

UNIVERZA V LJUBLJANI  
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anej Lekše

## **Avtomatizacija delavniškega dnevnika**

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM  
PRVE STOPNJE  
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: dr. Andrej Brodnik

Ljubljana, 2020

COPYRIGHT. Rezultati diplomske naloge so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavo in koriščenje rezultatov diplomske naloge je potrebno pisno privoljenje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

*Besedilo je oblikovano z urejevalnikom besedil L<sup>A</sup>T<sub>E</sub>X.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogu:

Tematika naloge:

Preveri, ali so glasovni asistenti v trenutnem stanju primerni za pomoč pri pisanju delavniških dnevnikov. Seznani se z obstoječimi programskimi rešitvami in jih analiziraj. Po analizi področja se loti izdelave svojega sistema za pisanje delavniških dnevnikov, ki vključuje glasovnega asistenta.



*Na tem mestu zapišite, komu se zahvaljujete za izdelavo diplomske naloge. Pazite, da ne boste koga pozabili. Utegnil vam bo zameriti. Temu se da izogniti tako, da celotno zahvalo izpustite.*



Kavi. Hvala.



# Kazalo

## Povzetek

## Abstract

<b>1</b>	<b>Uvod</b>	<b>1</b>
1.1	Opis domene raziskave . . . . .	1
1.2	Struktura diplomske naloge . . . . .	2
<b>2</b>	<b>Pregled problema in rešitve</b>	<b>3</b>
2.1	Problem . . . . .	3
2.2	Obstoječe rešitve . . . . .	4
<b>3</b>	<b>Načrtovanje in razvoj sistema za pisanje opisov tehnoloških postopkov</b>	<b>11</b>
3.1	Definicija funkcionalnosti . . . . .	11
3.2	Načrt . . . . .	13
3.3	Uporabljene tehnologije in programska oprema . . . . .	15
3.4	Strežnik . . . . .	17
3.5	Implementacija „Alexa Skill“ . . . . .	34
3.6	Implementacija mobilne aplikacije . . . . .	40
3.7	Testiranje sistema OpenReport . . . . .	52
3.8	Evalvacija funkcionalnosti . . . . .	55
<b>4</b>	<b>Zaključek</b>	<b>57</b>





# Seznam uporabljenih kratic

kratica	angleško	slovensko
<b>API</b>	Application Programming Interface	vmesnik za programiranje aplikacije
<b>AWS</b>	Amazon Web Services	Amazonove spletne storitve
<b>FIFO</b>	First In First Out	Prvi not, prvi ven
<b>HTML</b>	HyperText Markup Language	Označitveni jezik za hipertekst
<b>HTTP</b>	HyperText Transfer Protocol	Protokol za prenos hiperteksta
<b>HTTPS</b>	HyperText Transfer Protocol Secure	Protokol za prenos hiperteksta, varna različica?
<b>IP</b>	Internet Protocol	Internetni protokol
<b>JSON</b>	JavaScript Object Notation	JavaScript zapis objektov
<b>JWT</b>	JSON Web Token	JSON spletni žeton
<b>LIMS</b>	Laboratory Information Management System	laboratorijski sistem za uporabljjanje informacij
<b>MIT</b>	Massachusetts Institute of Technology	Tehnološki inštitut Massachussettsa
<b>MVVM</b>	Model View View-Model	Model Pogled Pogled-Model
<b>NLU</b>	Natural Language Understanding	Razumevanje naravnega govora
<b>SQS</b>	Simple Queue Service	preprosta vrstna storitev
<b>UI</b>	User Interface	uporabniški vmesnik
<b>URL</b>	Universal Resource Locator	Univerzalni lokator virov
<b>VUI</b>	Voice User Interface	glasovni uporabniški vmesnik
<b>XAML</b>	Extensible Application Markup Language	razširljiv aplikacijski označitveni? jezik

# Povzetek

**Naslov:** Avtomatizacija delavníškega dnevnika

**Avtor:** Anej Lekše

Diplomsko delo obravnava področje vpeljave komercialno dostopnih glasovnih asistentov v programske rešitve kot sredstvo za pohitritev uporabe. V nalogi raziščemo, kako v računalniški sistem vpeljemo glasovnega pomočnika in vpliv glasovnega pomočnika na uporabo sistema. Cilj je implementirati in preizkusiti računalniški sistem z glasovnim asistentom, s pomočjo katerega lahko narekujemo zapiske med delom. Te zapiske pa lahko naknadno urejamo s pomočjo mobilne aplikacije. Uporabili smo Amazon Alexa zaradi enostavne izdelave lastnih programov (t.i. „Skill-ov“). Ugotovili smo, da Amazon Alexa ni bila zmožna razpozнатi izgovorjenih stavkov z veliko variirajočimi besedami. Uspešna je bila pri razpoznavanju stavkov, ki so se v večini ali v celoti ujemali z naučenim razpoznavnim modelom. Kljub temu je čas, ki je potreben, da Alexa razpozna prejet glasovni ukaz prepočasen, da bi bil sistem učinkovit.

**Ključne besede:** mobilni razvoj, glasovni asistenti, razpoznavanje glasu, informacijski sistemi.



# Abstract

**Title:** Workshop report automatization

**Author:** Anej Lekše

This thesis deals with the process of incorporating a commercially available voice assistant into a software solution as a means of optimizing the workflow. The goal of thesis was implementing and testing a system with a voice assistant that could be used to write workshop report steps during work. These notes can then be edited with a mobile application. We have used Amazon Alexa, as it offers simple programming with Alexa Skills. We discovered that Amazon Alexa was unsuitable for literal dictation of longer, largely varying text. It was successful at recognising phrases that fully or mostly matched its learned model. Regardless, Alexa's ability to process custom voice commands, at the time of this research, was too slow to be deemed efficient.

**Keywords:** mobile development, voice assistants, voice recognition, information systems.



# Poglavlje 1

## Uvod

### 1.1 Opis domene raziskave

Diplomsko delo obravnava področje vpeljave glasovnih asistentov v programske rešitve kot sredstvo za izboljšanje učinkovitosti dela.

Za področje, ki ga želimo izboljšati, smo si izbrali pisanje delavniškega dnevnika ali opisa tehnološkega postopka. Kljub temu se bo naš sistem, z manjšimi spremembami, lahko uporabil tudi za pisanje drugačnih vrst poročil in navodil. Med te spadajo na primer kuharski recepti, laboratorijski dnevniki, itd.

Opis tehnološkega postopka ali delavniški dnevnik je dokument, ki po korakih predstavi postopek izdelave izdelka. Delavniški dnevniki imajo lahko definirane tudi kontrolne postopke za izdelek, orodje, ki ga rabimo za izdelavo, in seznam možnih nevarnosti pri delu.

Delavniški dnevnik je sestavljen iz zaporedja korakov, ki si sledijo v časovnem zaporedju. Vsak korak sestavlja opis postopka in predvideno trajanje. Koraki lahko vsebujejo tudi slikovne razlage.

Opisi tehnološkega postopka se najpogosteje uporabljajo v proizvodnih obratih tovarn kot navodila in oporne točke za sestavljalce.

V sklopu diplomske naloge smo želeli izdelati specializiran sistem za pisanje delavniških dnevnikov, nato pa preveriti, ali lahko z vpeljavo glasovnega

asistenta delo s tem sistemom pohitrimo. Ta sistem bi sestavljeni glasovni asistent, ki bi služil za narekovanje opomb, mobilna aplikacija, preko katere bi lahko urejali zapiske in strežnik, ki bi hrani podatke.

Raziskati želimo predpostavko, da lahko z učinkovito uporabo glasovnega asistenta zmanjšamo čas interakcije z mobilno aplikacijo.

## 1.2 Struktura diplomske naloge

Diplomsko delo pričenjam s predstavitvijo področja raziskave in kratko opišemo problem in možno rešitev. Postavimo si hipotezo.

Začnemo z raziskavo obstoječih rešitev za pisanje opisa tehnološkega postopka. Nato opišemo, kaj trenutne rešitve te problematike ponujajo in njihove prednosti ter slabosti.

V naslednjem poglavju se lotimo načrtovanja specializiranega sistema z glasovnim asistentom za pomoč pri pisanju delavnih dnevnikov. Nato se lotimo opisa tehnologij, ki smo jih pri pisanju diplome uporabili. Natančno definiramo funkcionalnosti sistema, utemeljimo odločitev za izbiro Amazon Alexa, AWS SQS, AWS Lambda in ogrodja Xamarin.

Nato predstavimo Alexa „Skill“, strežniški program in mobilno aplikacijo. Funkcionalnosti sistema testiramo in analiziramo.

V predzadnjem poglavju opišemo možnosti nadaljnega razvoja projekta.

# Poglavlje 2

## Pregled problema in rešitve

### 2.1 Problem

#### 2.1.1 Kaj vsebuje delavniški dnevnik?

Delavniški dnevnik je dokument, ki po korakih predstavi postopek izdelave izdelka. Delavniški dnevniki imajo lahko definirane tudi kontrolne postopke za izdelek, orodje, ki ga potrebujemo za izdelavo in seznam možnih nevarnosti pri delu.

Delavniški dnevnik je sestavljen iz zaporedja korakov, ki si sledijo v časovnem zaporedju. Vsak korak je sestavljata opis postopka dela in predvideno trajanje. Koraki lahko vsebujejo tudi slikovne razlage.

Delavniški dnevniki se najpogosteje uporabljam v proizvodnih obratih tovarn kot navodila in oporne točke za sestavljalce. V podjetju Gorenje takšne dokumente imenujejo opis tehnološkega postopka (slika 2.1). Lahko se uporabljam tudi kot učni pripomoček za spremeljanje napredka npr. dijakov vajencev na praksi [4].

A	B	C	D	E	F	G	H	I	K
Material	Kratki tks. mat.	Delovni Postopek	Kratki tekst postop.	Osnovna koef.	Stand.	Enota stand.	Stand.	Stand.	Stevilo zaposlenih
HLADIL		OV847P 0010	Operacija planiranja	1	1	S	41.9	0	0
HLADIL		OV847A 0020	sestavljalec2- trojček	1	0	S	47	3	3
HLADIL		OV660B 0030	kontrola ohisij	1	0	S	47	1	1
HLADIL		OV660A 0040	čiščenje po puru	1	0	S	47	1	3
HLADIL		OV847A 0050	sestavljalec 2	1	0	S	45	15	15
HLADIL		OV847A 0060	sestavljalec 1	1	0	S	45	30	30
HLADIL		OV847B 0070	strojnik embalirnega stroja	1	0	S	47	1	1
HLADIL		OV847B 0080	pregledovalec sklopov	1	0	S	47	6	6
HLADIL		OV847Q 0090	mehanik gospodinjskih aparatov	1	0	S	47	3	3
HLADIL		OV847B 0100	varilec gasflux, lak popravilo (1.5)	1	0	S	47	3	2
HLADIL		OV847S 0110	operacija brez delavca	1	27	S	47	0	0
HLADIL		OV847S 0120	vakuum krog	1	22	S	47	0	0
HLADIL		OV847S 0130	100% kontrola	1	19	S	47	0	0
HLADIL		OV847S 0140	povezava montaža - opremjanje	1	6	S	47	0	0
HLADIL		OV847A 0150	Vijačenje nosilca predala	1	0	S	47	1	1
HLADIL		OV847A 0160	Vstavljanje ventilatorja	1	0	S	47	1	1
HLADIL		OV847A 0170	Sestava profilov	1	0	S	47	1	1
HLADIL		OV847A 0180	Odstranjevanje folije	1	0	S	47	2	2
HLADIL		OV847P 0190	Sestava steklene police	1	0	S	19.7	1	4.1
HLADIL		OV847A 0200	Feniranje buširke	1	0	S	22.32	1	1
CELICA		OV784P 0010	Linija za navajanje uporjal - planira	1	1	S	140	0	0
CELICA		OV784S 0020	naviganje	1	1	S	140	0	0
CELICA		OV784O 0030	Pripraviti celico, naviti cev	1	0	S	140	1	1
CELICA		OV784A 0040	Spojiti celico in CPU	1	0	S	136	1	1
CELICA		OV784A 0050	He detekcija	1	0	S	140	1	1

Slika 2.1: V tabelo izvožen opis tehnološkega postopka za izdelavo hladilnika

// vir je gorenjje??

### 2.1.2 Definicija problematike

V tej diplomski nalogi raziščemo možne metode izdelave opisa tehnološkega postopka. Raziskati želimo prednosti in slabosti trenutnih metod. V naslednji fazi raziskave želimo izdelati specializiran sistem za sestavljanje opisov tehnoloških postopkov. Raziskati želimo, ali uporaba glasovnega asistenta v takšnem sistemu skrajša interakcije z mobilno aplikacijo.

Izhajajoč iz navedenega opredeljujemo problematiko diplomskega dela:

// TODO

## 2.2 Obstojče rešitve

### 2.2.1 Pisanje na papir

Najstarejša metoda za izdelavo takšnega dokumenta je zapis na list papirja (slika 2.2).

// TODO obreži sliko

## Opis tehnološkega postopka

Avtor: ANEJ LEKŠE

Datum: 18.8.2020

Naslov:	ČIŠĆENJE TIPKOVNICE
Opis:	ČIŠĆENJE MEHANSKE TIPKOVNICE
Nevarnosti:	/

Slika 2.2: Opis tehnološkega postopka za čiščenje tipkovnice na papirju.

Prednosti uporabe papirja in pisala so, da pri delu izdelovalec ne potrebuje računalnika in cenovna ugodnost. Slabosti takšnega postopka so:

- omejitve glede velikosti prostora, namenjenega vsakemu koraku,
- problematično dopisovanje in urejanje obstoječih korakov,
- zahtevnejše arhiviranje od računalniških datotek,
- občutljivost papirja na fizične poškodbe (trganje, mečkanje, vnetljivost,...).

### 2.2.2 Pisarniški programi

Opis tehnološkega postopka lahko izdelamo v pisarniških programih, na primer Microsoft Word ali LibreOffice Writer.

Ta pristop reši večino slabosti uporabe papirja in pisala za pisanje opisa tehnološkega postopka. Korake lahko enostavno dodajamo in urejamo. Prav tako je možno dodajati slikovno gradivo. Pisarniški programi omogočajo enostaven izpis dokumenta na tiskalnik, če želimo imeti dokument natisnjen na papirju.

Kljub temu uporaba te metode prinese nove slabosti:

- če imamo dokument shranjen na več mestih, moramo ob spremembah zagotoviti, da se spremenijo vse shranjene različice.
- Slikovno gradivo je vezano na dokument. Ob spremembah moramo spremeniti celoten dokument, ne le slike.
- Možnost izgube ali izbrisala podatkov.

V sklopu diplomske naloge smo se odločili napisati preprost opis tehnološkega postopka (slika 2.3) s programom LibreOffice Writer [8]. Enostavno ga je bilo izvoziti v format PDF. Dokument brez vsebine se lahko pri naslednjem opisu uporabi kot obrazec.

Opis tehnološkega postopka

Avtor: Anej Lekše

Datum: 12. 8. 2020

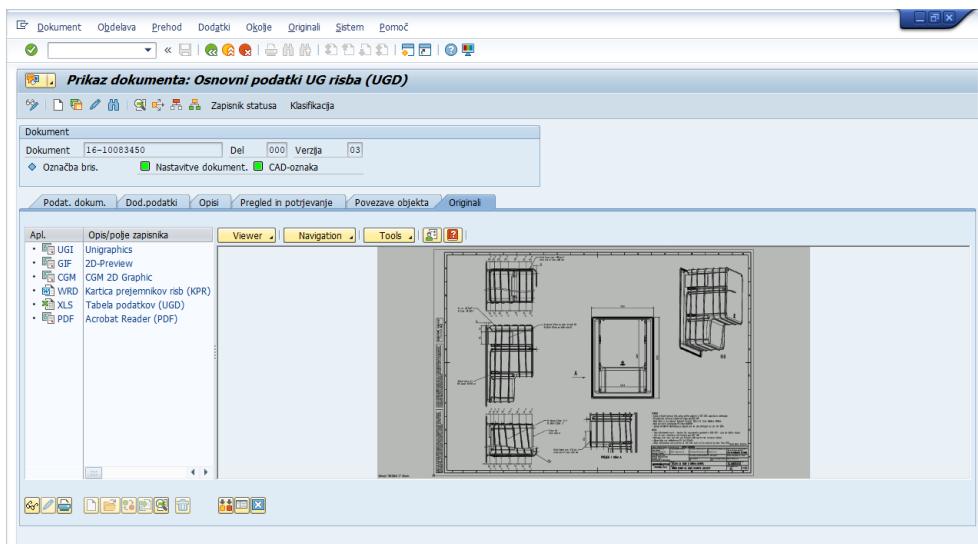
Naslov:	<b>Čiščenje tipkovnice</b>
Opis:	Čiščenje mehanske tipkovnice
Nevarnosti:	Ni nevarnosti

Korak	Opombe	Trajanje
Pripravi delovno mizo in tipkovnico		5 min
Odstrani pokrove tipk	Uporabi orodje, ki je bilo priloženo ob tipkovnici	10 min
Pokrove tipk namoči v milnico	Pusti v vodi vsaj 10 minut	10 min
Obriši pokrove tipk do suhega		30 min
Vstavi pokrove tipk na svoja mesta		8 min

Slika 2.3: Preprost opis tehnološkega postopka za čiščenje tipkovnice, napisan v pisarniškem programui.

### 2.2.3 Specializirani moduli za poslovne informacijske sisteme

Podjetja za svoje izdelke večinoma uporabljajo specializirane module za lastne informacijske sisteme. Primer je prikazan modul za informacijski sistem SAP (na sliki 2.4).



Slika 2.4: Prikazan korak opisa tehnološkega postopka s programom SAP.

Do tega opisa tehnološkega postopka se lahko dostopa iz računalnikov na delovnih mestih, kjer se izvajajo koraki, opisani v dokumentu. Opis tehnološkega postopka v sistemu SAP sestavlja:

- podatki o izdelku,
- dodatne opombe,
- opisi korakov,
- definicija kontrolnih postopkov in pregleda,
- CAD izris izdelka.

Specializiran informacijski sistem, kot je SAP, je tesno povezan s proizvodno linijo in prilagodljiv za potrebe proizvodnega obrata, ki ga uporablja. Pri sistemu SAP se podatki primarno hranijo na osrednjem strežniku, varnostna kopija pa se inkrementalno dela na geografsko ločen rezervni strežnik. Glavne slabosti takšnega sistema so:

- potrebna proizvodna infrastruktura, ki jo sistem rabi za optimalen izkoristek in

- cena, ki je potrebna za implementacijo takega sistema v proizvodno linijo.



# Poglavlje 3

## Načrtovanje in razvoj sistema za pisanje opisov tehnoloških postopkov

### 3.1 Definicija funkcionalnosti

Pri načrtovanju sistema za pisanje opisov tehnoloških postopkov smo se osredotočali predvsem na nasledje točke:

- prekinitve dela, ki jih povzroči uporaba sistema morajo biti čim krajše,
- uporabniški vmesnik odjemalca mora omogočati karseda učinkovito delo,
- komponente sistema morajo biti enostavno zamenljive,
- uporabljeni podatki (fotografije, zapiske) se mora hrani na enem mestu,
- sistem mora biti popolnoma funkcionalen brez glasovnega asistenta.

Sistem, ki smo ga implementirali v tej raziskavi, služi kot primer kompleksnejšega sistema, katerega uporabo bi radi pospešili z uporabo glasovnega asistenta.

Sistem bo služil kot alternativa pisanju opisov tehnoloških postopkov s pisarniškimi programi. Sistem mora podpirati ustvarjanje novega opisa tehnološkega postopka in odpiranje ter urejanje obstoječih opisov tehnološkega postopka.

Vsak opis tehnološkega postopka mora imeti naslov, opis, opis možnih nevarnosti pri delu in seznam korakov dela.

Seznam korakov dela mora podpirati dodajanje novih korakov, urejanje obstoječih korakov, brisanje obstoječih korakov in spreminjanje vrstnega reda korakov.

### 3.1.1 Kaj smo želeli z glasovnim asistentom izboljšati?

Z glasovnim asistentom smo poskusili izboljšati dodajanje korakov. To smo poskusili z dodajanjem dobesedno narekovanih korakov.

Poleg narekovanja korakov smo poskusili z glasovnimi ukazi nadzirati mobilno aplikacijo. Z glasovnimi ukazi smo nadzirali odpiranje obrazcev za dodajanje korakov in zagon kamere.

Glasovni asistent v našem sistemu ima naslednje vloge:

- za trenutno odprt opis tehnološkega postopka je z njim možno dodajati korake postopka,
- z njim je možno odpreti kamero ali obrazec za dodajanje koraka.

Raziskava iz leta 2018 [3] je pokazala izboljšano učinkovitost pri delu raziskovalcev v kemijskem laboratoriju, v katerega so integrirali glasovne pomočnike. Namenska raziskava je bil preizkus praktične uporabnosti glasovnih asistentov za branje laboratorijskih postopkov in glasovno upravljanje laboratorijskih instrumentov. Pozitivni rezultati bi lahko bili ključnega pomena za slabovidne člane laboratorijev. Kot glasovni asistent je bila uporabljeni Amazon Alexa. Prepoznavanje govora in ukazov je bilo konsistentno in hitro, ne glede na spol uporabnika. Motnje pri razpoznavanju je povzročal večinoma ozadni hrup. Povprečna uspešnost prepoznavane ukazov je bila 95%.

Raziskovalci so zabeležili tudi problem moteče kakofonije v laboratoriju, v katerem je več raziskovalcev, ki uporabljajo glasovni nadzor naprav.

### 3.2 Načrt

Sistem za pisanje opisov tehnoloških postopkov smo poimenovali OpenReport. Sistem OpenReport sestavlja:

- strežniški program,
- mobilna aplikacija,
- glasovni asistent.

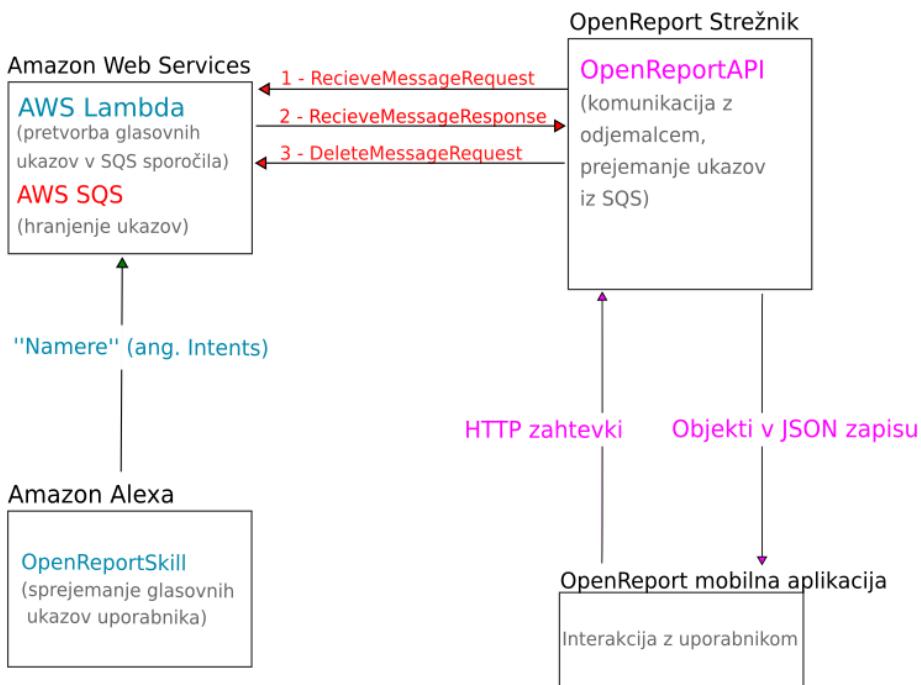
Strežnik v podatkovni bazi hrani uporabnike, delavnische dnevnike in korake. Uporabnik lahko do podatkov na strežniku dostopa preko klienta, ki s strežnikom komunicira preko API. V našem primeru je to mobilna aplikacija. Glasovni asistent smo uporabili kot dodatek k mobilni aplikaciji. Omogoča glasovno upravljanje aplikacije in narekovanje korakov.

Preko mobilne aplikacije uporabnik lahko:

- opravi registracijo in prijavo,
- ustvari nov delavnishi dnevnik,
- odpre obstoječe delavnische dnevnike,
- ustvarja in ureja korake delavnishkega dnevnika,
- zajema slike in jih vstavlja v delavnishi dnevnik,
- briše korake delavnishkega dnevnika,
- ureja vrstni red korakov delavnishkega dnevnika.

Za glasovni asistent Amazon Alexa smo razvili „Skill“, s katerim uporabnik lahko:

- v odprto poročilo vstavi dobesedno narekovan korak,
- odpre obrazec za dodajanje novega tekstovnega koraka,
- odpre kamero in obrazec za dodajanje koraka s fotografijo.



Slika 3.1: Visokonivojski načrt sistema

### 3.2.1 Zasnova aplikacije in vpliv na hipotezo

Mobilna aplikacija bo služila kot primaren način za interakcijo s strežnikom.

V tej diplomski nalogi raziskujemo, ali je možno z vpeljavo glasovnega asistenta v naš sistem izboljšati učinkovitost tega sistema. Da bodo rezultati raziskave čim bolj natančni, bomo dali učinkovitosti mobilne aplikacije velik poudarek. Uporabniški vmesnik bo zasnovan za čim hitrejše delo. To nam bo v zagotovilo, da uporabniški vmesnik aplikacije ni "ozko grlo" pri pisanju opisov tehnoloških postopkov.

Če bo delo z aplikacijo kljub dodelanemu in učinkovitemu uporabniškemu vmesniku hitrejše, bomo hipotezo potrdili. V nasprotnem primeru bomo hipotezo ovrgli.

### 3.3 Uporabljene tehnologije in programska oprema

#### 3.3.1 .NET

.NET [1] je razvojna platforma, razvita s strani Microsofta. Obsega programske jezike, prevajalnike, orodja in knjižnice, ki omogočajo širok spekter primerov uporabnosti, hkrati pa se ohranja enovitost ozadne kode.

Tehnologije .NET ogrodja, ki smo jih uporabil v tej diplomski nalogi so:

- .NET Core - odprtokodna platforma za razvoj spletnih storitev,
- Xamarin - ogrodje za razvoj mobilnih aplikacij za najpogostejše mobilne operacijske sisteme (Android, iOS).

.NET smo izbrali, saj je zelo dobro integriran z Amazonovim AWS API-jem in je dobro dokumentiran.

#### 3.3.2 Xamarin

Ogrodje Xamarin [12] je odprtokodno orodje za razvoj mobilnih aplikacij, ki ga je razvil Microsoft.

Z ogrodjem Xamarin je mogoče pri deliti večino ozadne in ospredne kode med različnimi mobilnimi operacijskimi sistemi. Za ozadno kodo se uporablja .NET (C#), za front-end pa se uporablja XAML (Extensible Application Markup Language).

Xamarin smo izbrali, da bomo lahko strežniški program in mobilno aplikacijo programirali v enakem programskem jeziku (C#).

### 3.3.3 Amazon Alexa

Amazon Alexa je glasovni asistent, razvit s strani podjetja Amazon [2]. Za Amazon Alexo smo se odločili, saj ponuja enostavno možnost programiranja s „Skill-i“. Poleg tega je Alexa enostavno integrirati z drugimi Amazonovimi spletnimi storitvami, ki jih bomo uporabili v tej diplomski nalogi (AWS SQS, AWS Lambda).

### 3.3.4 Alexa Skill

Alexine osnovne funkcionalnosti lahko nadgradimo s programi, ki se jim reče „Skill“ [10]. Da lahko ustvarimo in objavimo „Skill“ rabimo račun Amazon razvijalca (*ang. Amazon Developer Account*).

„Skill“ sestavlja:

- **Invocation** - fraza, ki „Skill“ zažene,
- **Intent** - fraze, ki jih „Skill“ razpozna kot funkcije,
- **Endpoint** - omrežni vir, kjer se nahaja ozadna koda „Skill-a“.

Ko Alexa zasliši ”Invocation“ ali katerega od ”Intent-ov“, glasovni posnetek pošlje na Amazonov strežnik. Ta s pomočjo glasovnega razpoznavnega modela prepozna uporabnikove ukaze in pošlje poseben zahtevek na ”Endpoint“. To je lahko druga Amazonova storitev (npr. AWS Lambda), storitev na Microsoftovem Azure strežniku ali naš lasten strežnik, dostopen preko javne domene.

Ko ”Endpoint“ obdela zahtevo, se odgovor pošlje nazaj na Amazonov strežnik v obliki znakovnega niza. Ta podatek se nato pošlje nazaj na uporabnikovo Alexo, ki prejeti znakovni niz ”izgovori“.

### 3.3.5 Amazon Web Services

AWS je skupek oblačnih storitev, ki jih ponuja podjetje Amazon. Ponuja integracijo s pogostimi programskimi jeziki in ogrodji, kot so Java, .NET,

Python in Node.js preko storitve AWS API. Pri tej diplomski nalogi smo se osredotočili na dva sistema iz skupka AWS.

### AWS SQS

AWS SQS je sistem za pošiljanje tekstovnih sporočil med odjemalci preko Amazonovih strežnikov [6]. Za hranjenje sporočil je treba registrirati ”SQS Queue” (*slo. SQS vrsto*). Ta je lahko neurejena vrsta, kjer prejeta sporočila niso nujno urejena po času ustvarjanja, lahko pa je tipa FIFO, pri kateri je zagotovljeno, da dobimo vsa sporočila v vrstnem redu, v katerem so bila poslana. V sklopu te diplome smo uporabili vrsto FIFO.

To storitev bomo uporabili za komunikacijo med Amazon Alexa in OpenReport strežnikom.

### AWS Lambda

Ozadno kodo za Alexa „Skill“ smo gostili na platformi AWS Lambda. To je storitev za gostovanje dogodkovno vodene ozadne kode [5]. Za to platformo smo se odločili zaradi dobre integracije z ”Alexa Skill Kit-om” in razvojnim orodjem Visual Studio.

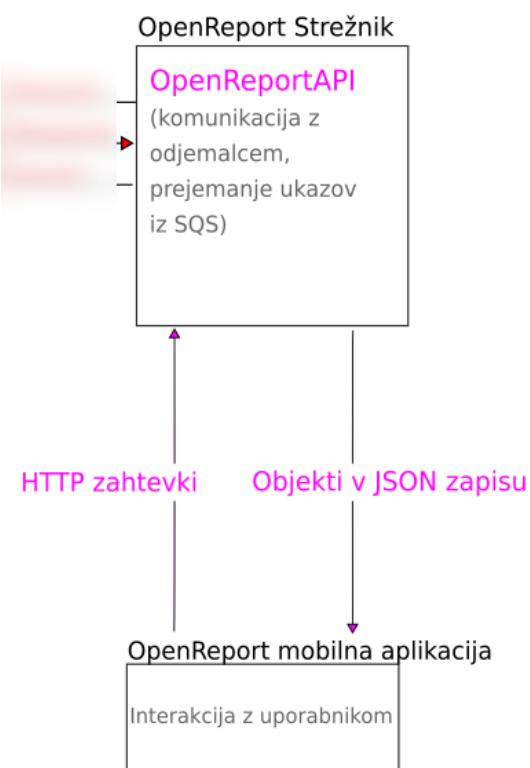
## 3.4 Strežnik

Za uporabo strežnika smo se odločili, da lahko do hranjenih podatkov dostopamo iz različnih naprav preko enotnega vmesnika (API). V našem primeru bosta s strežnikom komunicirala mobilna aplikacija in glasovni asistent. Podatke bomo hranili v podatkovni bazi, ki jo streže SQL Server.

### 3.4.1 Komunikacija s klienti

Strežnik z odjemalci komunicira preko API. To metodo smo izbrali zaradi enostavnosti implementacije in možnosti širjenja nabora odjemalcev v prihodnosti.

V našem primeru bo edini odjemalec mobilna aplikacija. Odjemalec na definirane funkcjske URL-je (slika 3.4) strežnika pošlje HTTP zahtevke. Če so zahtevki pravilno formulirani, strežnik izvede predvideno funkcijo in rezultat te funkcije pošlje kot HTTP odgovor nazaj odjemalcu. Komunikacija med strežnikom in odjemalcem je prikazana z roza barvo na spodnji sliki (slika 3.2).

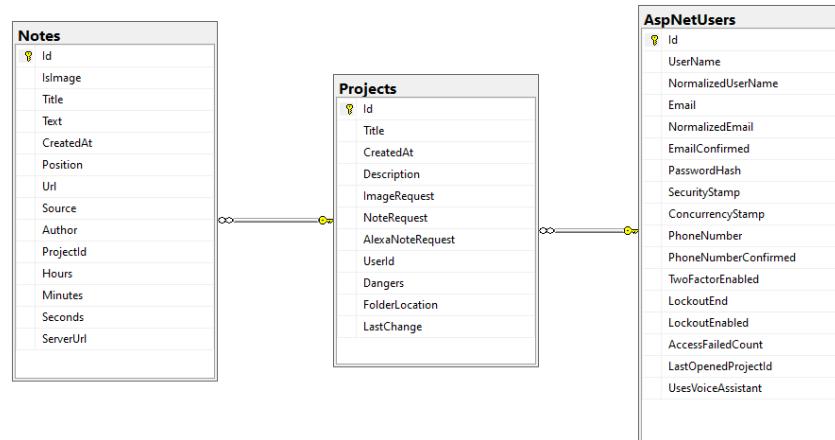


Slika 3.2: Komunikacija strežnika s klientom

### 3.4.2 Podatkovna baza

Podatkovna baza hrani tabele ”Users” (*slo. Uporabniki*), ”Projects” (*slo. Projekti*) in ”Notes” (*slo. Zapiski*) (slika 3.3). Vsak uporabnik lahko ima 0 ali mnogo projektov (opisov tehnološkega postopka). Vsak projekt ima lahko

0 ali mnogo zapiskov (korakov).



Slika 3.3: ER model podatkovne baze

### 3.4.3 Implementacija API

Komunikacija med strežnikom in odjemalcji poteka preko HTTP po pristopu API (slika 3.4). Podatki v HTTP sporočilih se zapisejo v format JSON.

Ker lahko sistem uporablja več uporabnikov, smo se odločili implementirati sistema za avtentifikacijo uporabnikov in avtorizacijo zahtev. Avtentikacija z uporabniškim imenom in gesлом omogoča preverjanje identitete. Ko se vzpostavi zaupanje, se prijavljenemu uporabniku dodeli avtorizacijski žeton. Iz žetona je možno razbrati, za katerega uporabnika gre in katere pravice so mu dodeljene.

**Connect**

GET /api/v1/connect

**Identity**

POST /api/v1/identity/register

POST /api/v1/identity/login

**Project**

GET /api/v1/projects

POST /api/v1/projects

GET /api/v1/projects/{id}

DELETE /api/v1/projects/{id}

GET /api/v1/projects/{id}/notes

GET /api/v1/projects/{id}/nonotes

POST /api/v1/projects/{id}/addnote

POST /api/v1/projects/{id}/addimage

POST /api/v1/projects/{id}/export/text

POST /api/v1/projects/{id}/export/html

PUT /api/v1/projects/last/closerequests

PUT /api/v1/projects/{pid}/update/{nid}

PUT /api/v1/projects/{pid}/{nid}/{positions}

DELETE /api/v1/projects/{pid}/delete/{nid}

**Users**

PUT /api/v1/users/requestava

PUT /api/v1/users/releaseava

POST /api/v1/users/current

Slika 3.4: Avtomatsko dokumentiranje funkcijskih URL-jev s Swagger

## Registracija in prijava

Pri registraciji mora odjemalec na strežnik poslati objekt razreda *RegisterUserRequest*. Odjemalec ga mora poslati na URL ”/identity/register” preko metode POST. V objektu *RegisterUserRequest* se nahajata njegov e-mail naslov in geslo.

```
RegisterUserRequest {  
    string Email;  
    string Password;  
}
```

Ko strežnik prejme ta objekt, ga pošlje v avtentikacijsko storitev. V tej storitvi preveri, ali uporabnik s tem uporabniškim imenom že obstaja v bazi (je že bil registriran). Če obstaja, se zabeleži napaka in nadaljnja registracija se prekine. Če ta uporabnik ne obstaja, se kreira nov objekt razreda *User*. Polje *Password* se zakriptira in se skupaj s poljem *Email* zapise v ta objekt. Objekt se nato zapise v podatkovno bazo v tabelo *Users*. Uporabnik pri registraciji dobi tudi svoj unikaten identifikator *UserId*.

V kolikor se je tekom avtentikacije dogodila kakršna koli napaka, se klientu vrne objekt razreda *AuthFailedResponse*. V tem objektu se odjemalcu pošlje seznam vseh napak, ki jih je avtentikacijska storitev zabeležila pri neuspešni registraciji.

```
AuthFailedResponse {  
    IEnumerable<string> Errors;  
}
```

Če je bila registracija uspešna, se odjemalcu pošlje objekt razreda *AuthSuccessResponse*.

```
AuthSuccessResponse {  
    string UserId;  
    string Token;  
}
```

V polje `Token` razreda `AuthSuccessResponse` se zapiše avtorizacijski žeton. Žeton je tipa JWT ali *JSON Web Token*. Sestavlja ga e-mail uporabnika, uporabnikov unikatni identifikator `UserID`, čas zapada žetona in tip sime-tričnega kodiranja, uporabljenega za enkripcijo žetona.

Objekt `AuthSuccessResponse` se nato pošlje nazaj odjemalcu. Odjemalec nato žeton iz prejetega objekta doda glavi vseh svojih HTTP zahtevkov na OpenReport strežnik. Strežnik ga uporabi pri poizvedbah po podatkovni bazi.

Prijava poteka podobno. Uporabnik mora poslati na strežnik objekt `LoginUserRequest`. Ciljni URL za ta zahtevek je ”/identity/login”. V objektu `LoginUserRequest` se nahajata njegov e-mail naslov in geslo.

```
(LoginUserRequest {  
    string Email;  
    string Password;  
})
```

Ko strežnik prejme ta objekt, ga pošlje v avtentikacijsko storitev. V tej storitvi preveri, ali uporabnik s tem e-mail naslovom že obstaja. Če uporabnik ne obstaja, ali pa je njegovo zakriptirano geslo v podatkovni bazi drugačno kot to, kar je v polju `Password`, se zabeležijo napake in prijava se prekine. Odjemalec prejme objekt razreda `AuthFailedResponse`.

Če uporabnik obstaja in se njegovo zakriptirano geslo iz polja `Password` ujema z gesлом v podatkovni bazi, je avtentikacija uspešna. Odjemalec prejme objekt razreda `AuthFailedResponse`.

V primeru uspešne prijave se avtorizacijski žeton generira in pošlje enako kot pri registraciji.

## Operacije z delavniki

Vsak uporabnik lahko ima nič ali več delavniskih dnevnikov.

Vse operacije nad uporabnikovim delavniskim dnevnikom morajo biti avtorizirane. V kolikor niso, bo API vedno javil napako `Bad Request: Not`

**Authorised.**

Pri ustvarjanju novega opisa tehnološkega postopka (v nadaljevanju po-glavja *projekta*) mora uporabnik podati naslov, kratek opis projekta in seznam možnih nevarnosti.

Te podatke odjemalec zapiše v objekt razreda `CreateProjectRequest`.

```
CreateProjectRequest {  
    string Title;  
    string Description;  
    string Dangers;  
}
```

Ta objekt se nato pošlje preko POST metode na strežnik na naslov `/projects/create`. Storitev za operacije nad projekti nato kreira nov objekt razreda `Project` s podatki iz prejete zahteve. V primeru, da je zahteva ustrezno formuirana in avtorizirana, se projekt zapiše v podatkovno bazo. Odjemalcu se kot odgovor pošlje ustvarjen objekt tega projekta.

Razred `Project` izgleda tako:

```
Project {  
    int Id; // unikatni identifikator  
    string Title;  
    string Description;  
    string Dangers;  
    IEnumerable<Note> Notes; // seznam korakov  
    ...  
}
```

Do kateregakoli svojega projekta lahko odjemalec v nadaljnji komunikaciji dostopa tako, da pošlje avtoriziran GET zahtevek na URL `"/projects/{id}"`. Polje `{id}` mora biti unikatni identifikator projekta.

Za izbris projekta mora uporabnik poslati DELETE zahtevo na URL `"/projects/{id}"`. Če zahteva ni avtorizirana z ustreznim žetonom, se projekt ne izbriše.

## Operacije nad koraki opisa tehnološkega postopka

Vsak projekt ima lahko nič ali več korakov. Vsak korak ima naslov, opis in predvideno trajanje.

Korak je lahko izključno tekstovni, lahko pa ima tudi pripadajočo sliko.

Da uporabnik ustvari tekstovni korak, mora poslati objekt razreda `Note`, preko POST metode, na URL ”/projects/{id}/addnote”. Polje `{id}` mora biti unikatni identifikator projekta, kateremu želi dodati korak.

Razred `Note` izgleda tako.

```
Note {  
    ...  
    string Title;  
    string Text;  
    int Hours;  
    int Minutes;  
    int Seconds;  
    ...  
}
```

Strežnik najprej preveri, če je pošiljatelj poslal avtorizirano zahtevo. Iz avtorizacijskega žetona strežnik razbere uporabnikov `UserID`. Če je projekt z identifikatorjem `{id}` res v lasti uporabnika z identifikatorjem `UserID`, se bo korak dodal v zbirko korakov tega projekta.

Če želi uporabnik ustvariti slikovni korak, mora poslati objekt razreda `UploadImageRequest` na URL ”/projects/{id}/addimage”.

```
UploadImageRequest {  
    Note note;  
    string ImageString;  
}
```

V tem razredu predstavlja polje `ImageString` pripadajočo fotografijo, zakodirano v znakovni niz. Za kodirni algoritem smo uporabili Base64.

Za hranjenje tekstovnih in slikovnih korakov smo zaradi preprostosti implementacije uporabili isti razred (**Note**). Tekstovni in slikovni korak se ločita v vrednosti boolean zastavice **IsImage**. Tekstovni korak ima to zastavico nastavljeno na vrednost **false**, slikovni pa **true**.

```
Note {  
    bool IsImage;  
    string Title;  
    string Text;  
    int Hours;  
    int Minutes;  
    int Seconds;  
    string Url; // Lokacija slike na klientu  
    string ServerUrl; // Lokacija slike na strežniku  
    ...  
}
```

Poljema **Url** in **ServerUrl** se dodeli vrednost samo pri slikovnih korakih. Ko se na klientu zajame fotografija, klient nastavi vrednost polja **Url** na lokacijo ustvarjene fotografije v datotečnem sistemu.

Polje **ServerUrl** se nastavi šele na strežniku. Ko strežnik prejme zahtevo za kreiranje slikovnega koraka, se korak **Note** prebere iz objekta **UploadImageRequest** in zapiše v podatkovno bazo. Nato se **ImageString** dekodira v slikovno datoteko in hrani na strežniku. Ko se fotografija uspešno zapiše v datotečni sistem, se v polje **ServerUrl** zapiše lokacija pravkar ustvarjene datoteke. Spremembe se hranijo v podatkovni bazi.

### Urejanje in brisanje in korakov v projektu

Pri izbrisu koraka iz projekta, mora uporabnik poslati avtorizirano zahtevo na URL ”/projects/{pid}/delete/{nid}”. V tem primeru je polje **{pid}** unikatni identifikator projekta, v katerem se nahaja korak, **{nid}** pa unikatni identifikator objekta **Note**, ki ga želimo izbrisati. V kolikor ima ta objekt

`Note` postavljeno zastavico `IsImage` na `true`, se poleg zapisa v bazi izbriše tudi pripadajoča slikovna datoteka.

Pri posodabljanju (urejanju) tekstovnih korakov, mora uporabnik poslati avtorizirano PUT zahtevo na URL ”/projects/{pid}/update/{nid}”. Telo zahteve mora vsebovati objekt razreda `Note`.

Pri posodabljanju slikovnih korakov mora uporabnik poslati avtorizirano PUT zahtevo na naslov ”/projects/{pid}/update/{nid}/image”. Telo te zahteve mora vsebovati objekt razreda `UploadImageRequest`. V objektu `UploadImageRequest` mora biti polje `Note` objekt, ki ga želimo posodobiti. Polje `ImageString` mora biti zakodirana slika, ki bo zamenjala prejšnjo sliko. Slika mora biti zakodirana po metodi Base64.

### Spreminjanje vrstnega reda korakov v projektu

Položaj koraka `Note` v projektu lahko razberemo iz atributa `Position`. Prvi korak ima `Position` 0, drugi 1, itd.

```
Note {  
    int Id;  
    int Position;  
    bool IsImage;  
    string Title;  
    string Text;  
    int Hours;  
    int Minutes;  
    int Seconds;  
    string Url;  
    string ServerUrl;  
}
```

Pri dodajanju korakov v projekt se obstoječe korake projekta razvrsti po vrednosti polja `Position`. Najdemo največjo vrednost tega polja, ki se v seznamu pojavi in ji prištejemo 1. Nato to vrednost priredimo polju `Position`

novo kreiranega koraka.

Ko uporabnik želi spremeniti pozicijo koraka v projektu, moramo poslati zahtevo PUT na URL ”/projects/{pid}/{nid}/{positions}”.

Polje {pid} predstavlja unikatni identifikator projekta, v katerem se nahaja korak. Polje {nid} predstavlja unikatni identifikator koraka Note. Polje {positions} pa prestavlja število mest za kolikor ga želimo prestaviti. To število je lahko pozitivno ali negativno celo število. Negativna vrednost prestavi korak proti začetku seznama, pozitivna pa proti koncu.

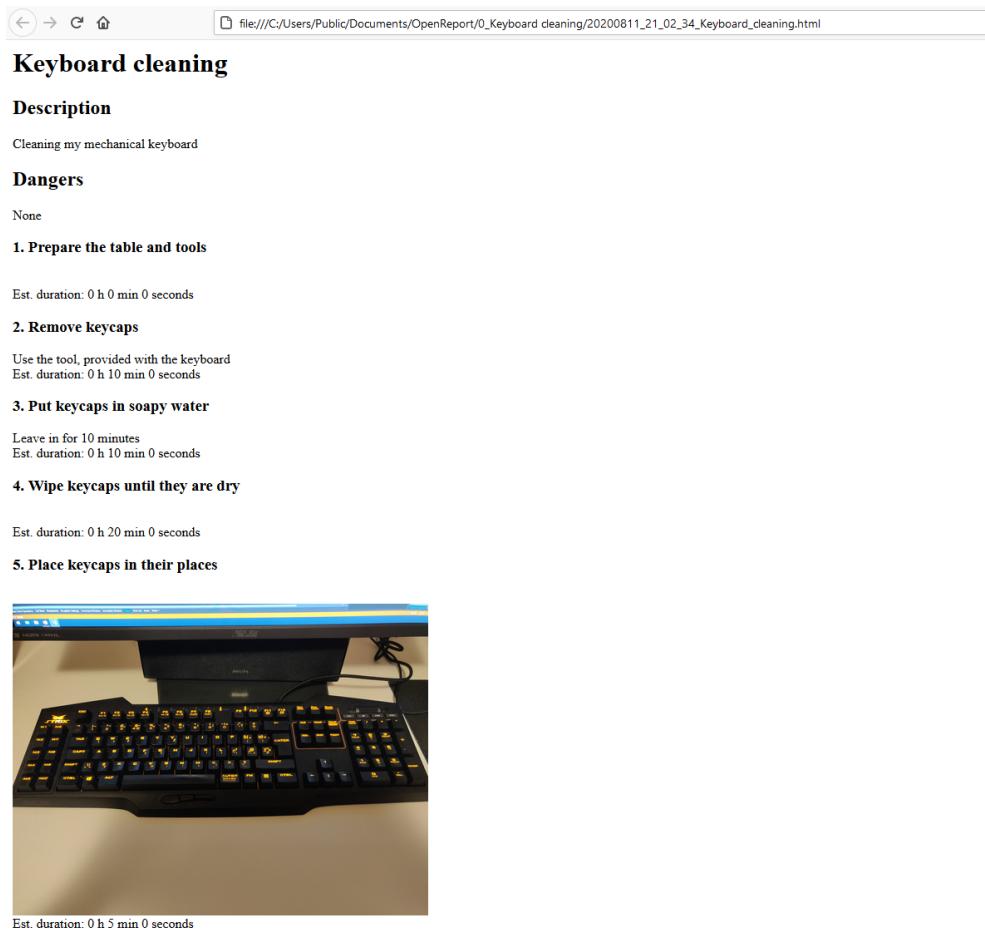
## Izvoz projektov

Projekt lahko uporabnik izvozi v dva formata, v tekstovno datoteko in HTML dokument.

V tekstovno datoteko ga uporabnik izvozi, tako, da pošlje avtorizirano GET zahtevo na URL ”/projects/{id}/export/text”. V HTML dokument ga uporabnik izvozi, tako, da pošlje avtorizirano GET zahtevo na URL ”/projects/{id}/export/html”.

Pri izvozu v tekstovno datoteko, storitev za upravljanje projektov na strežniku ustvari novo tekstovno datoteko. Najprej vanjo vpiše naslov in opis projekta ter možne nevarnosti pri delu. Nato uredi korake po vrednosti stolpca Position in besedila korakov eno za drugim zapiše v datoteko. Pri slikovnih korakih se, poleg besedila, zapiše tudi lokacija pripadajoče fotografije.

Pri izvozu v HTML dokument (slika 3.5) se naslov projekta zapiše kot HTML naslov H1, opis projekta kot naslov H2, koraki pa kot HTML odstavki. V HTML dokumentu lahko prikažemo poleg besedila slikovnih korakov tudi slike.

A screenshot of a web browser window displaying a cleaning guide for a keyboard. The title is "Keyboard cleaning". The guide includes sections for "Description" (Cleaning my mechanical keyboard), "Dangers" (None), and five steps: 1. Prepare the table and tools (Est. duration: 0 h 0 min 0 seconds), 2. Remove keycaps (Use the tool, provided with the keyboard, Est. duration: 0 h 10 min 0 seconds), 3. Put keycaps in soapy water (Leave in for 10 minutes, Est. duration: 0 h 10 min 0 seconds), 4. Wipe keycaps until they are dry (Est. duration: 0 h 20 min 0 seconds), and 5. Place keycaps in their places (Est. duration: 0 h 5 min 0 seconds). Below the steps is a photograph of a black mechanical keyboard.

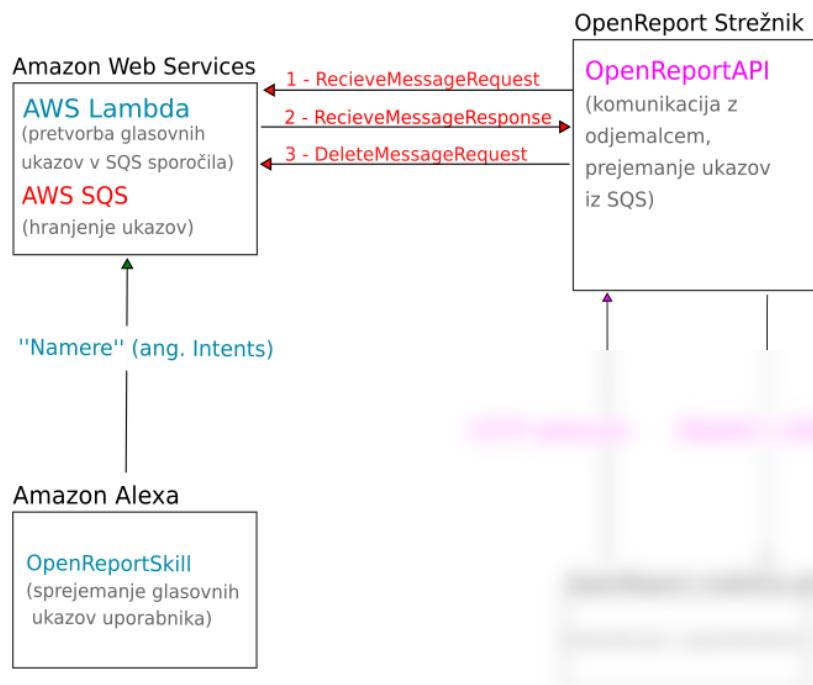
Slika 3.5: Projekt, izvožen v HTML dokument

Lokacija izvožene datoteke se nahaja v polju `FolderLocation` v razredu `Project`. Privzeta lokacija, ki se nastavi ob kreiranju projekta je `"C:/users/$USER/Public Documents/OpenReport"`.

```
Project {
    int Id;
    string Title;
    string Description;
    string Dangers;
    string FolderLocation; // lokacija pripadajočih datotek
```

```
IEnumerable<Note> Notes;
...
}
```

### 3.4.4 Komunikacija strežnika z glasovnim asistentom



Slika 3.6: Komunikacija med strežnikom in glasovnim asistentom preko SQS

Strežnik z glasovnim asistentom Amazon Alexa komunicira preko storitve AWS SQS (slika 3.6). Glasovni asistent od uporabnika prejme glasovni ukaz. Na podlagi tega ukaza, glasovni asistent na SQS vrsto odloži tri različna tekstovna sporočila:

- ”addnote”,
- ”addimage”,

- ”{narekovano besedilo}”,

Strežnik periodično prejema sporočila iz SQS vrste. Ko sporočilo prejme, na podlgi njegove vsebine izvede eno od treh možnih funkcij.

- Sporočilo ”addnote” sporoči strežniku, naj na odjemalcu odpre obrazec za dodajanje tekstovnega koraka.
- Sporočilo ”addimage” sporoči strežniku, naj na odjemalcu odpre kamerino in obrazec za dodajanje slikovnega koraka.
- Sporočilo ”{narekovano besedilo}” sporoči strežniku, naj vsebino tega sporočila doda odprtemu projektu kot tekstovni korak.

V sporočilu ”{narekovano besedilo}” se nahaja besedilo, ki ga je glasovni asistent razpoznał kot dobesedni narek koraka. Primer takšnega sporočila je ”this is a literal dictation” ali ”prepare the table”.

## Utemeljitev uporabe SQS

Prenos sporočil med Amazon Alexa in strežnikom smo realizirali z uporabo storitve AWS SQS. Pri uporabi SQS je prenos podatkov resda manj učinkovit kot pri direktni povezavi preko spletne vtičnice (*ang. Web Socket*).

Glavni faktor, zaradi katerega smo se odločili za uporabo AWS SQS, je, da z dodatno plastjo za hranjenje sporočil, dosežemo enostavno zamenljivost glasovnega asistenta.

Pri morebitni menjavi glasovnega asistenta, strežniškega programa ne bi spreminjałi. Potrebno bi bilo le programirati novega glasovnega asistenta, da na AWS SQS odlaga sporočila v strežniku razumljivem formatu (”addnote”, ”addimage”,...).

## Zahevanje in izpust glasovnega asistenta

Na enem OpenReport strežniku bomo uporabljali samo enega glasovnega asistenta. Tega asistenta lahko uporabnik ”zahteva” za lastno uporabo in ga po

uporabi "izpusti". Če ima določen uporabnik nase vezanega asistenta, vidimo po vrednosti zastavice `UsesVoiceAssistant`. Ta se nahaja v uporabnikovem objektu razreda `User`. Uporabnik asistenta uporablja, če je zastavica nastavljena na `True`.

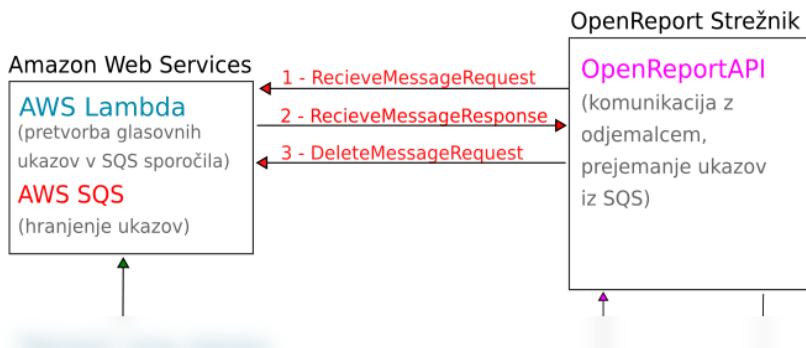
Ko uporabnik odpre katerega od svojih projektov, se identifikator tega projekta nastavi v polje `LastOpenedProjectId` v njegovem objektu razreda `User`.

Zahtevki, ki jih strežniku pošlje Alexa preko SQS vrste, se navezujejo na projekt s tem identifikatorjem.

```
User : IdentityUser {  
    string Id;  
    string Email;  
    string Password;  
    IEnumerable<Project> Projects;  
    ...  
    int LastUsedProjectId;  
    bool UsesVoiceAssistant;  
}
```

## Delovanje

Komunikacija med strežnikom in glasovnim asistentom je prikazana z rdečo barvo na spodnji sliki (slika 3.7).



Slika 3.7: Komunikacija med strežnikom in SQS

Na strežniku OpenReport teče storitev za komunikacijo z glasovnim asistentom. Ta storitev vsakih pet sekund na SQS pošlje zahtevek `RecieveMessageRequest`.

```

RecieveMessageRequest {
    AttributeName
    MaxNumberOfMessages
    QueueUrl
    WaitTimeSeconds
    ...
}
    
```

AWS SQS kot odgovor vrne objekt tipa `RecieveMessageResponse`.

```

RecieveMessageResponse {
    IEnumerable<Message> Messages;
    ...
}
    
```

V tem odgovoru se nahaja seznam sporočil, ki čakajo na sprejem iz SQS vrste. V kolikor je v odgovoru vsaj eno sporočilo, pregledamo telo vseh sporočil.

Ko strežnik sporočila sprejme, jih mora izbrisati iz SQS vrste. Če jih sproti ne izbriše, iz FIFO vrste ne more brati najnovejših sporočil, ampak le 5 najstarejših.

Sporočila izbriše iz SQS vrste z zahtevkom `DeleteMessageRequest`. Ta zahtevek hrani referenco na sporočilo, ki ga želimo izbrisati iz SQS vrste. AWS SQS storitev v odgovor vrne `DeleteMessageResponse`, a v tej diplomski nalogi tega odgovora ne uporabimo več.

Zahtevki, ki jih strežnik prejme od glasovnega asistenta se odražajo na zadnjem projektu, ki ga uporabnik z glasovnim asistentom odpre.

- Zahtevek ”`addnote`” nastavi zastavico ”`NoteRequest`” na `true`,
- zahtevek ”`addimage`” nastavi zastavico ”`ImageRequest`” na `true`,
- zahtevek ”`{narekovano besedilo}`” nastavi zastavico ”`AlexaNoteRequest`” na `true` in prejeto besedilo doda projektu kot korak.

```
Project {
    int Id;
    string Title;
    string Description;
    string Dangers;
    string FolderLocation;
    IEnumerable<Note> Notes;
    bool NoteRequest;
    bool ImageRequest;
    bool AlexaNoteRequest;
}
```

Če je vsebina prejetega sporočila enaka `addnote`, storitev preveri, kateri uporabnik ima trenutno nase vezanega glasovnega asistenta. Nato v projektu, ki ga je nazadnje odprl, nastavi vrednost zastavice `NoteRequest` na `true`.

Če je vsebina enaka ”`addimage`”, storitev preveri, kateri uporabnik ima trenutno nase vezanega glasovnega asistenta. Nato v projektu, ki ga je nazadnje odprl, nastavi vrednost zastavice `ImageRequest` na `true`.

Če vsebina prejetega sporočila ni enaka ”`addimage`” ali ”`addnote`”, pomeni, da je prejeto sporočilo dobesedno narekovan korak. Kot prej, storitev preveri, kateri uporabnik ima nase vezanega glasovnega asistenta. Nato v projektu, ki je nazadnje odprl, nastavi vrednost zastavice `AlexaNoteRequest` na `true`. Poleg tega ustvari nov objekt razreda `Note`, ki ima naslov ”Voice note”, vsebina pa je telo prejetega sporočila. Nov objekt se nato vstavi v zadnji odprt projekt.

### 3.5 Implementacija „Alexa Skill-a“

Alexa „Skill“ smo želeli uporabiti kot sredstvo za pohitritev dela z mobilno aplikacijo. Glavna funkcija Alexa „Skill-a“ v sklopu sistema OpenReport je dodajanje korakov v opis tehnološkega postopka in odpiranje obrazcev za vstavljanje korakov.

Natančneje, „Skill“ uporabniku omogoča:

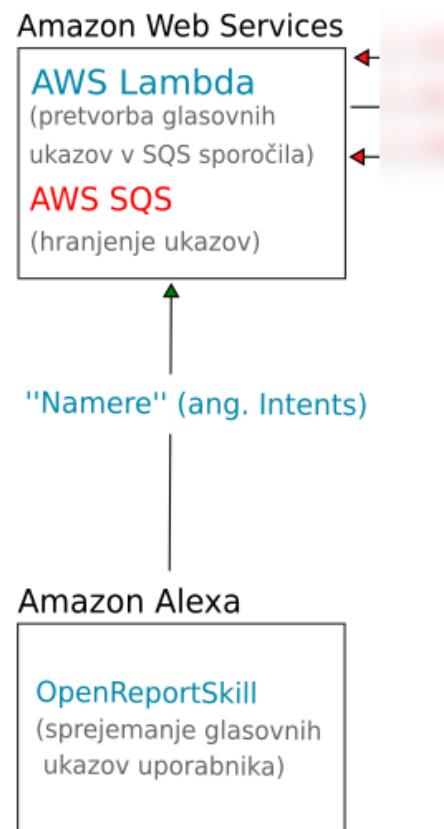
- dodajanje tekstovnega koraka,
- odpiranje kamere in dodajanje slikovnega koraka in
- dobesedno narekovanje besedila tekstovnega koraka.

Ko Alexa zasliši ukaz za zagon „Skill-a“ (ang. Invocation), se „Skill“ začne izvajati. Invokacijska fraza se glasi ”*make a report note*“.

Vsak nadaljnji ukaz, ki ga uporabnik izreče, preden se „Skill“ preneha izvajati, se bo primerjal s frazami za zagon funkcij „Skill-a“. Te fraze se imenujejo ”namere“ (ang. Intent) (slika 3.8). V naš „Skill“ smo implementirali tri glavne namere:

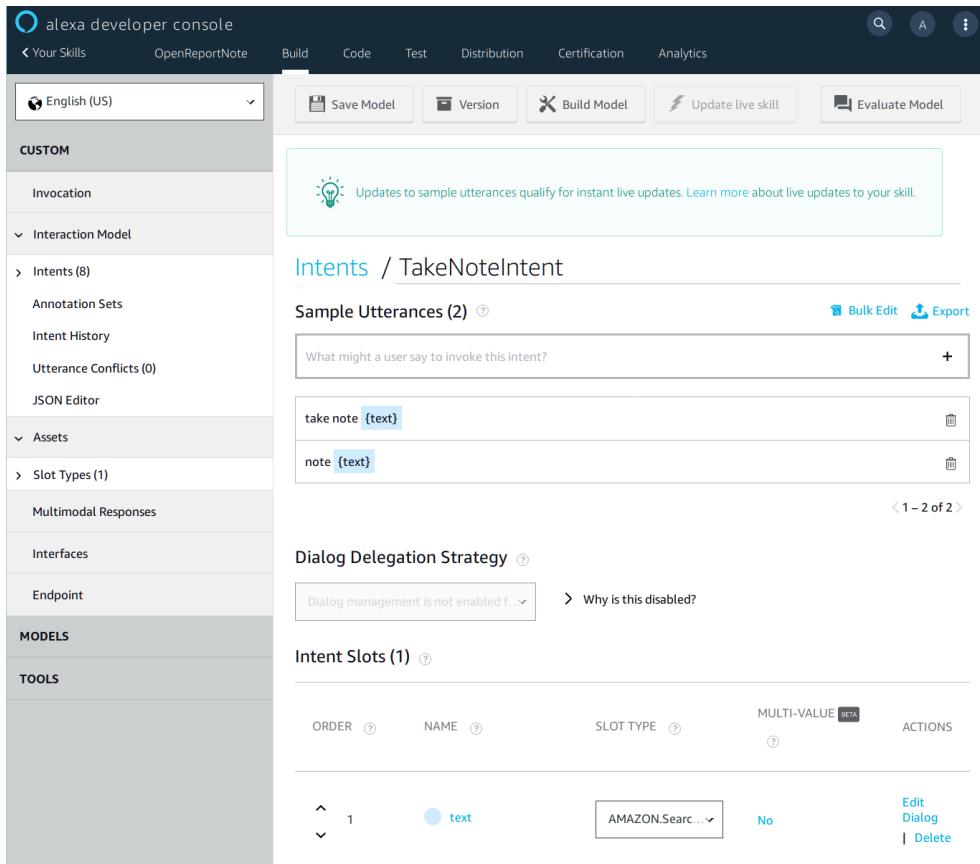
- `TakeNoteIntent`,

- `OpenTextNoteFormIntent`,
- `OpenImageNoteFormIntent`.



Slika 3.8: Načrt komunikacije Alexe, AWS Lambda in AWS SQS

`TakeNoteIntent` (slika 3.9) se zažene, ko po invokacijski frazi uporabnik izreče `"take note {besedilo}"` ali `"note {besedilo}"`. Polje `{besedilo}` je tekstovna spremenljivka, v katero „Skill“ hrani razpoznano tekstovno vsebino koraka. V primeru, da uporabnik izreče `"note unscrew the backplate"`, je vrednost spremenljivke `{besedilo}` `"unscrew the backplate"`.



Slika 3.9: TakeNoteIntent v Alexa Developer nadzorni plošči

TakeNoteIntent v ozadni kodi „Skill-a“ zažene funkcijo, ki bo ustvarila novo SQS sporočilo. V telo tega sporočila funkcija vstavi vrednost spremenljivke {besedilo} in ga pošlje v SQS vrsto. „Skill“ zatem vrne uporabniku odgovor ”Noted!” in se preneha izvajati.

OpenTextNoteFormIntent se zažene, ko po invokacijski frazi uporabnik izreče ”create a text note”.

OpenTextNoteFormIntent v ozadni kodi „Skill-a“ zažene funkcijo, ki ustvari novo SQS sporočilo. V telo tega sporočila funkcija vstavi besedilo ”addnote” in ga pošlje v SQS vrsto. „Skill“ zatem vrne uporabniku odgovor ”Opening the form!” in se preneha izvajati.

---

[Intents / OpenTextNoteFormIntent](#)Sample Utterances (2) ?

What might a user say to invoke this intent?

create a text note

add a text note

Slika 3.10: OpenTextNoteFormIntent

`OpenImageNoteFormIntent` se zažene, ko po invokacijski frazi uporabnik izreče `"take a picture"`.

Ta "Intent" v ozadni kodi „Skill-a“ zažene funkcijo, ki ustvari novo SQS sporočilo. V telo tega sporočila funkcija vstavi besedilo `"addimage"` in ga pošlje v SQS vrsto. „Skill“ zatem vrne uporabniku odgovor `"Launching camera!"` in se preneha izvajati.

---

[Intents / OpenImageNoteFormIntent](#)Sample Utterances (1) ?

What might a user say to invoke this intent?

take a picture

Slika 3.11: OpenImageNoteFormIntent

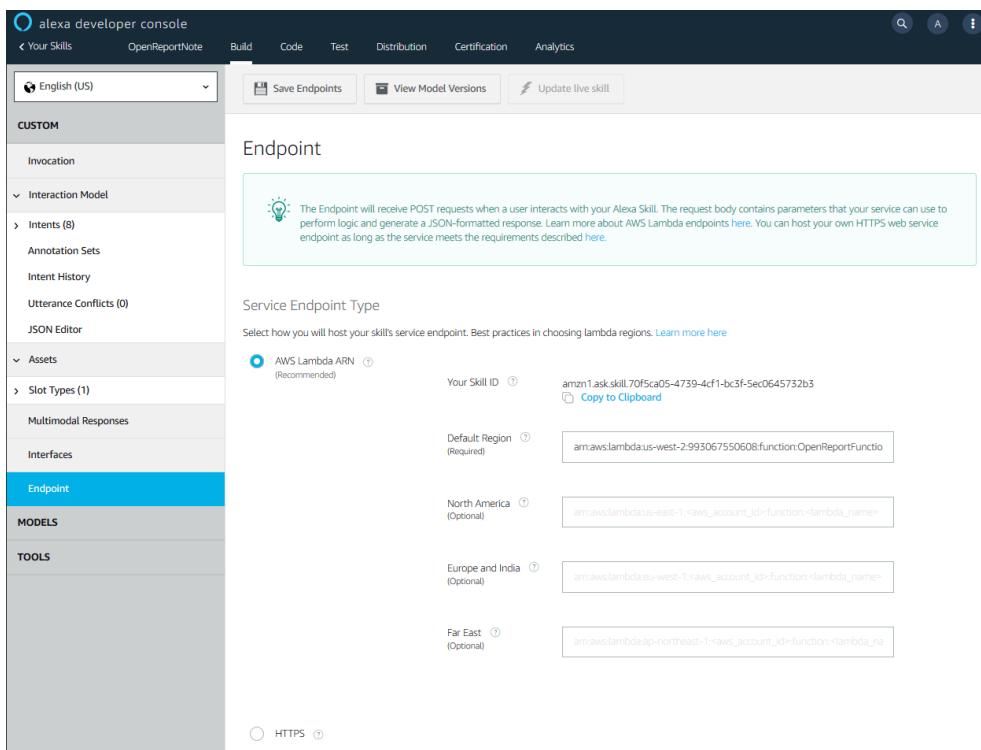
### 3.5.1 Izvajanje „Skill-a“ po korakih

Alexa „Skill“ se začne izvajati, ko uporabnik izreče definirano invokacijsko frazo. Alexa posnet govorni ukaz pošlje na Amazonov Alexa Server. Tam se s pomočjo NLU razpoznavnega modela poskusi pretvoriti v znakovni niz.

Ta znakovni niz se primerja z vsemi definiranimi invokacijskimi frazami za „Skill-e“, ki so vezani na naš Amazon račun.

Če Alexa Server uporabnikov glasovni ukaz razpozna kot „make a report note“, se začne izvajati OpenReportAlexaSkill. V ozadno kodo „Skill-a“ (ang. Endpoint) se pošlje zahteva tipa LaunchRequest.

”Endpoint“ nastavimo v Alexa Developer nadzorni plošči (slika 3.12). ”Endpoint“ je lahko AWS Lambda funkcija, lahko pa je naš lasten strežnik, dostopen preko javne domene. ”Endpoint“ je v našem primeru gostovan na storitvi AWS Lambda. Ob prejetju LaunchRequest ”Endpoint“ našega „Skill-a“ vrne vprašanje ”What now?“.



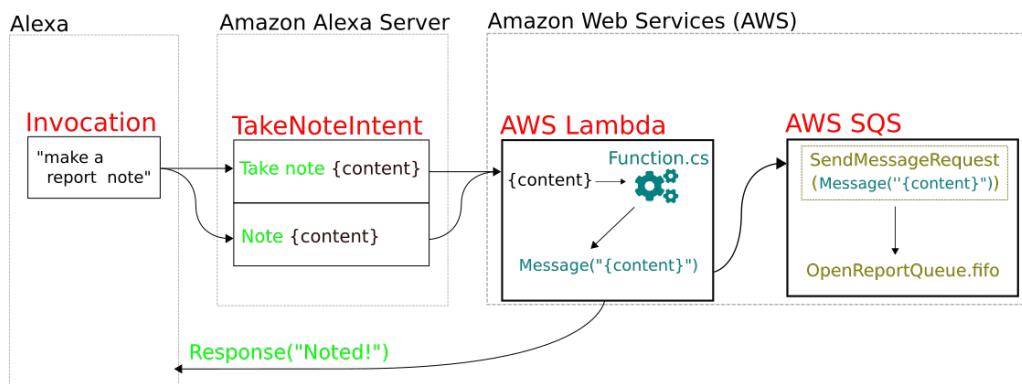
Slika 3.12: ”Endpoint“ OpenReportAlexaSkill-a v Alexa Developer nadzorni plošči

Naslednji glasovni ukaz, po zagonu „Skill-a“, se primerja s frazami za namere (ang. Intent). Če se ukaz ujema z frazami, ki zaženejo katerega od

”Intent-ov“ TakeNoteIntent, OpenTextNoteFormIntent ali OpenImageNoteFormIntent, se na ”Endpoint“ pošlje zahteva tipa IntentRequest.

Iz prejetega IntentRequest-a nato dobimo ime ”Intent-a“. Na podlagi imena prejetega ”Intent-a“ ločimo, ali gre za TakeNoteIntent, OpenTextNoteFormIntent ali OpenImageNoteFormIntent in zaženemo funkcije, opisane v prejšnjem poglavju.

Oris obdelave invokacije „Skill-a“ in namere TakeNoteIntent je prikazan s spodnjo sliko (slika 3.13).



Slika 3.13: Načrt Alexa Skill-a

### 3.5.2 Ovire pri izdelavi Alexa „Skill-a“

Pri implementaciji Alexa „Skill-a“ smo naleteli na največ težav v sklopu te raziskave.

Prva je bila jezikovne narave, saj Alexa ne podpira prepoznavanja slovenskega jezika. Zato smo se odločili za uporabo angleškega jezika.

Naslednja težava je bila definicija invokacijske fraze. Fraze, kot so ”take a note“ ali ”take a picture“ so že rezervirane v sklopu Alexinih privzetih funkcionalnosti. To pomeni, da jih za naš „Skill“ ne moremo uporabiti.

Izogibati smo se morali tudi frazam, ki so bile rezerviranim frazam podobne (npr. ”open report note“ itd.). Tudi take fraze so se velikokrat razpoznale kot rezervirane fraze.

Težavo je predstavljalo izbrati frazo, je bila hkrati kratka, intuitivna in se ni napačno interpretirala v invokacijsko frazo privzetih funkcionalnosti. Invokacija "make a report note" je bila najboljši kompromis, ki smo ga našli.

Težavo je predstavljalo tudi razpoznavanje dolgih stavkov, še posebej, če se ti niso ujemali z definiranimi frazami namer. To je zelo omejilo možnost narekovanja besedila korakov s pomočjo Amazon Alexe.

Naslednjo težavo je predstavljal čas, ki ga je Alexa porabila za interpretacijo definiranih glasovnih ukazov. Za procesiranje glasovnih ukazov je porabila v povprečju od 3 do 6 sekund.

## 3.6 Implementacija mobilne aplikacije

Mobilno aplikacijo smo uporabili kot primer odjemalca za naš strežnik (slika 3.2).

Do strežnika dostopa preko API vmesnika. Zahteveki se prenašajo preko HTTP protokola.

Mobilno aplikacijo smo se odločili razviti s tehnologijo Xamarin. Razlogi za to so predvsem naše dobro poznavanje tehnologije in enostavna integracija z ostalimi tehnologijami iz .NET sklopa.

### 3.6.1 Pristop

Pristop razvoja aplikacije, ki smo ga uporabili za programiranje aplikacije, se imenuje "Model View View-Model" (v nadaljevanju MVVM). Pri tem pristopu aplikacijo razdelimo na tri dele (slika 3.14).

**Model** je del, kjer definiramo elemente naše "poslovne logike" (opis tehnološkega postopka, korak, uporabnik,...).

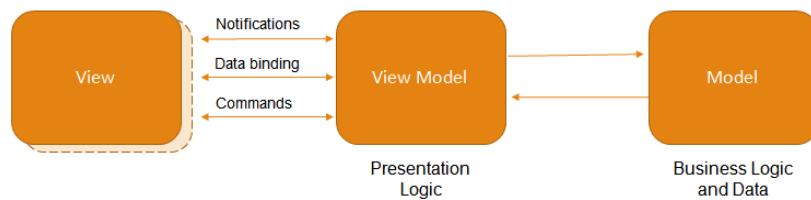
**View** je uporabniški vmesnik, ki ga vidi uporabnik.

**ViewModel** pa se uporablja, da se poveže funkcije uporabniškega vmesnika in modela ("poslovne logike") ter po potrebi preoblikuje podatke.

Rezultat upoštevanja tega pristopa je čista koda, ki nima prepleteneih elementov med ozadno kodo in uporabniškim vmesnikom. "Model" vse-

buje le abstrakcijo naših podatkov in "poslovno logiko". Ti podatki se v "ViewModel-u" pretvorijo v obliko, ki bo prikazana uporabniku. "View" pa nato servira pripravljene podatke uporabniku v obliki grafičnih elementov.

Na spodnji sliki lahko vidimo oris pristopa MVVM (slika 3.14) [9].

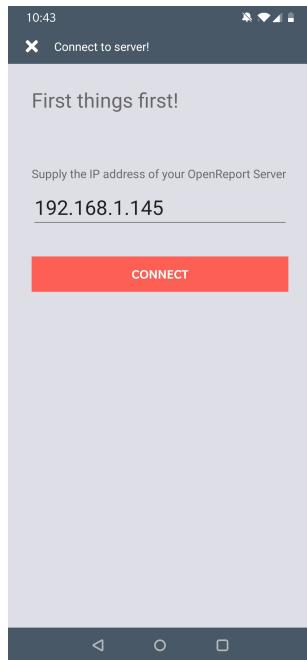


Slika 3.14: Shema pristopa MVVM

### 3.6.2 Povezava na strežnik, prijava in registracija uporabnika

Za kreiranje in avtorizacijo HTTP zahtevkov v okviru aplikacije skrbi storitev za komunikacijo s strežnikom (razred `OpenReportServerCommunicationService.cs`). Storitev se registrira ob zagonu aplikacije po metodi Dependency Injection. Za to metodo registracije storitev smo se odločili, saj omogoča enostavno uporabo instanc registriranih storitev kjerkoli v aplikaciji.

Ob zagonu mobilne aplikacije, uporabnika pričaka stran za povezavo na OpenReport strežnik. V tekstovno polje mora vpisati IP naslov strežnika. Ta naslov bomo v nadaljevanju podpoglavlja označevali z **naslovstrežnika**.



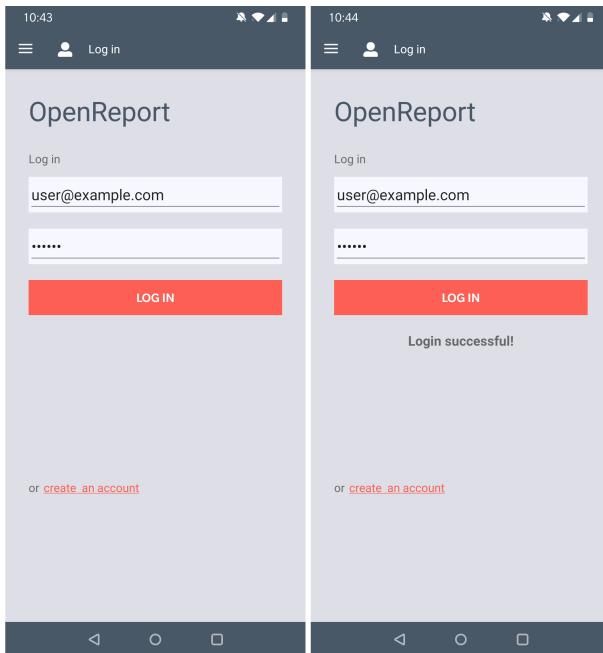
Slika 3.15: Povezavna stran

Ko uporabnik pritisne na gumb "Connect", se preveri, ali je naslov pravilno formuliran. Če je, storitev za komunikacijo s strežnikom pošlje neavtorizirano GET zahtevo na "**naslovstrežnika/connect**". Če je na tem naslovu res OpenReport strežnik, se kot odgovor odjemalcu pošlje boolean vrednost **true**.

Storitev za komunikacijo s strežnikom si nato hrani vrednost **naslovstrežnika** kot predpono vseh nadaljnjih zahtevkov.

Ob neuspešnem poskusu povezave, se uporabniku na strani prikaže opis napake. Če je poskus uspešen, se stran za povezavo na strežnik zapre.

Prikaže se stran za prijavo. Na tej strani uporabnik vpiše svoje uporabniško ime ali geslo, lahko pa odpre stran za registracijo računa.



Slika 3.16: Prijavna stran pred poskusom prijave (levo), in po uspešnem poskusu prijave (desno)

Za prijavo in registracijo skrbi avtentikacijska storitev (razred `IdentityService.cs`). Ta storitev je, enako kot storitev za komunikacijo s strežnikom, registrirana po metodi Dependency Injection. Uporabljamo jo lahko, ko je storitev za komunikacijo uspešno vzpostavila komunikacijo s strežnikom.

Poskus prijave poteka tako, avtentikacijska storitev uporabniško ime in geslo zapiše v objekt razreda `LoginUserRequest`. Uporabniško ime in geslo se vneseta na prijavni strani.

```
(LoginUserRequest {
    string Email;
    string Password;
})
```

Nato se objekt `LoginUserRequest` pošlje na strežnik. Odjemalec ga pošlje na URL ”/identity/login” preko POST metode. Strežnik ob uspešni

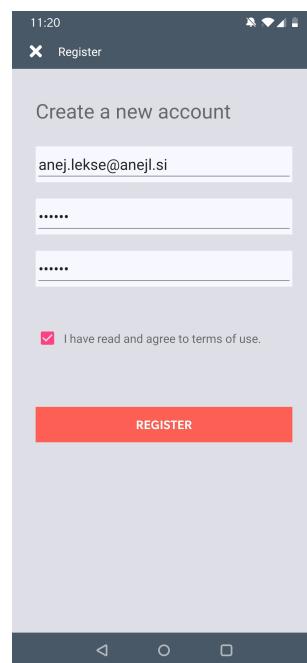
avtentikaciji vrne objekt razreda `AuthSuccessResponse`. Iz tega objekta uporabnik prebere svoj avtorizacijski žeton. Ta žeton hrani avtentikacijska storitev. Komunikacijska storitev ta žeton doda v glavo nadalnjih HTTP zahtevkov.

Če je avtentikacija neuspešna, strežnik vrne objekt razreda `AuthFailedResponse`, v katerem je podan opis napake. Ta opis napake se nato uporabniku izpiše na prijavni strani.

Stran za registracijo (slika 3.17) je podobna strani za prijavo, le, da ima dve vnosni polji za vpis gesla. Vrednost v obeh poljih za geslo mora biti enaka, da se lahko uporabnik registrira.

Registracijski zahtevek se pošlje na strežnik na URL ”/identity/register”. Zahtevek je objekt razreda `RegisterUserRequest`.

```
RegisterUserRequest {  
    string Email;  
    string Password;  
}
```



Slika 3.17: Registracijska stran

### 3.6.3 Ustvarjanje, brisanje in odpiranje projektov

Po prijavi lahko uporabnik odpre stran s projekti. Na tej strani se prikažejo:

- seznam vseh njegovih projektov,
- gumbi za izbris posameznega projekta,
- gumb za dodajanje novega projekta in
- gumb za zahtevanje in izpust glasovnega asistenta.

#### Ustvarjanje novega projekta

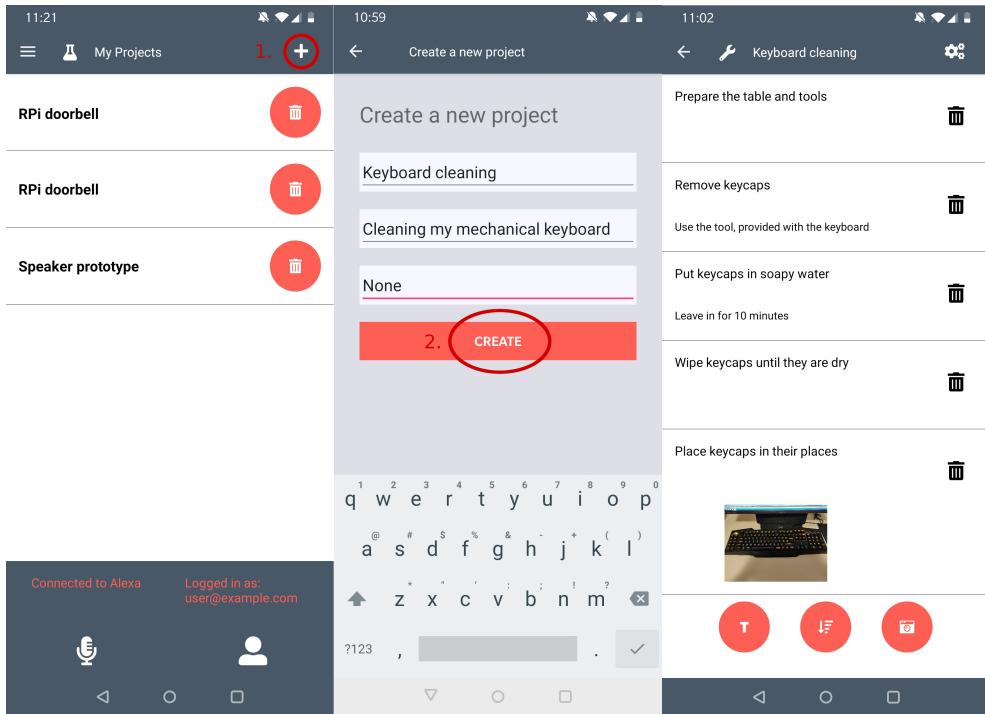
Za ustvarjanje novega projekta (slika 3.18), mora uporabnik odpreti stran za dodajanje projektov. Na tej strani mora vpisati naslov projekta (**Title**), kratek opis (**Description**) in našteti možne nevarnosti pri delu **Dangers**.

Te tri vrednosti se zapisa v objekt razreda `CreateProjectRequest`.

```
CreateProjectRequest {
    string Title;
    string Description;
    string Dangers;
}
```

Ta objekt nato storitev za komunikacijo s strežnikom pošlje preko metode POST na URL ”/projects/create”. Strežnik v odgovor vrne objekt razreda Project, ki ga je zahtevek ustvaril.

Nov projekt se nato prikaže v seznamu vseh uporabnikovih projektov na strani ”My Projects”, kjer ga lahko uporabnik odpre ali izbriše.



Slika 3.18: Postopek ustvarjanja novega projekta po korakih.

## Izbris projekta

Da uporabnik izbriše projekt, mora na strani ”My projects” ob željenem projektu pritisniti gumb z ikono smetnjaka. Pritisak na ta gumb sporoči sto-

ritvi za komunikacijo s strežnikom, naj pošlje avtoriziran DELETE zahtevek na URL ”/projects/{id}/delete”. Strežnik ob uspešnem izbrisu projekta odjemalcu vrne kopijo izbrisanega projekta.

### 3.6.4 Zahtevanje in izpust glasovnega asistenta

Storitev za upravljanje z glasovnim asistentom (`VoiceAssistantService.cs`) skrbi za zahtevanje in izpust glasovnega asistenta. Ta storitev se registrira ob začetku izvajanja aplikacije, nato pa počaka, da avtentifikacijska storitev uspešno opravi prijavo uporabnika. Ob prijavi uporabnika preveri, ali ta uporablja glasovnega asistenta. To doseže tako, da pošlje GET zahtevek na URL ”/users/current”. Strežnik vrne objekt `User`, ki pripada prijavljenemu uporabniku. Storitev preveri vrednost boolean atributa `UsesVoiceAssistant`.

Če ima uporabnik vrednost polja `UsesVoiceAssistant` nastavljeno na `true`, lahko ima glasovnega asistenta vezanega nase.

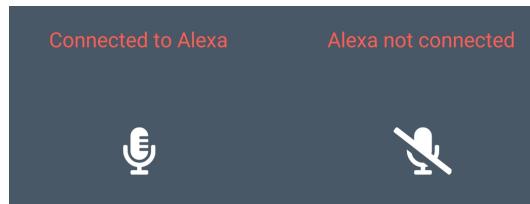
Če ima uporabnik vrednost polja `UsesVoiceAssistant` nastavljeno na `false`, mora uporabo glasovnega asistenta zahtevati. To naredi tako, da pošlje avtoriziran PUT zahtevek na URL ”/users/requestva”.

Če noben drug prijavljen uporabnik nima polja `UsesVoiceAssistant` nastavljenega na `true`, se njegova zahteva odobri. V njegovem objektu `User` se nastavi vrednost `UsesVoiceAssistant` na `true`.

Če ima kateri od uporabnikov to polje nastavljeno na `true`, zahteva ne bo odobrena. V tem primeru mora uporabnik, ki zahteva asistenta počakati, da uporabnik, ki asistenta uporablja, asistenta izpusti. Ko si asistenta ne lasti nobeden drug uporabnik, ga lahko veže nase.

Izpust asistenta dosežemo, da uporabnik, ki asistenta trentuno uporablja, pošlje avtoriziran PUT zahtevek na URL ”/users/releaseva”.

V mobilni aplikaciji zahtevanje in izpust asistenta nadziramo na ”Dashboard” strani, s pritiskom na ikono mikrofona (slika 3.19).



Slika 3.19: Gumb v obeh stanjih

### 3.6.5 Odpiranje in sinhronizacija projekta

Ko uporabnik iz seznama projektov izbere projekt, ki ga želi odpreti, storitev za komunikacijo s strežnikom pošlje na URL ”/projects/{id}/notes” zahtevek GET. Polje {id} je unikatni identifikator želenega projekta.

Storitev za projekte (razred `ProjectService.cs`) nato prejet odgovor, razreda `Project`, hrani. Vse korake, ki pripadajo temu projektu, se hrani v seznam tipa `ObservableCollection`. Ta tip seznama smo izbrali, saj je v ogrodju Xamarin nad njim najlažje opravljati zamenjevalne operacije. Seznam se nato prikažem na strani za upravljanje s projektom.

Storitev nato začne s periodično sinhronizacijo projekta. Storitev za projekte na URL ”/projects/{id}/nonotes” pošlje avtoriziran GET zahtevek. Strežnik vrne objekt razreda `Project` s praznim seznamom korakov. Vse, kar storitev za projekte pri sinhronizaciji gleda, je vrednost zastavic `NoteRequest`, `ImageRequest` in `AlexaNoteRequest`.

Na `true` nastavljena zastavica `NoteRequest` odpre obrazec za dodajanje novega tekstovnega koraka v projekt.

Na `true` nastavljena zastavica `ImageRequest` odpre obrazec za dodajanje nove fotografije v projekt.

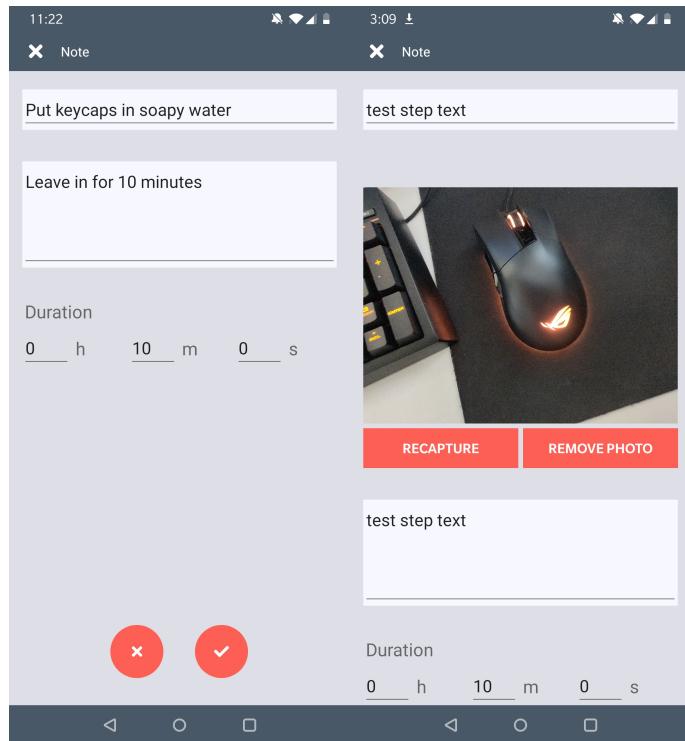
Na `true` nastavljena zastavica `AlexaNoteRequest` storitvi za projekte sporoči, naj pošlje GET zahtevek na URL ”/projects/{id}/notes”. Polje {id} mora biti unikatni identifikator odprtrega projekta. V odgovor prejme objekt razreda `Project`, vključno s seznamom vseh korakov, ki mu pripadajo. V prejetem seznamu je namreč tudi narekovan korak. Storitev za projekte

primerja, katerega od korakov še nima v svoji zbirki. Ko ta korak najde, ga doda v svoj ObservableCollection.

Ko se projekt sinhronizira, storitev za projekte pošlje na strežnik PUT zahtevo na URL ”/project/last/closerequests”. Ta zahteva nastavi vrednost zastavic NoteRequest, ImageRequest in AlexaNoteRequest na false.

### 3.6.6 Dodajanje in brisanje zapiskov

Ko ima uporabnik projekt odprt, lahko vanj dodaja korake, jih briše, ureja in jim spreminja vrstni red. Doda lahko tekstovni korak ali korak s fotografijo.



Slika 3.20: Stran za dodajanje/urejanje korakov

Pri dodajanju tekstovnega koraka mora uporabnik v vnosnem obrazcu (slika 3.20) podati naslov koraka (Title), opis koraka (Text) ter trajanje v urah, minutah in sekundah. Ti podatki se nato hranijo v objektu razreda

**Note.** Ta objekt storitev za projekte vključi v objekt razreda `AddNoteRequest` in ga pošlje storitvi za komunikacijo s strežnikom.

```
AddNoteRequest {  
    Note note;  
}
```

Storitev za komunikacijo s strežnikom pošlje ta objekt na strežnik. Pošlje ga na URL ”/projects/{id}/addnote” preko metode POST. Polje `{id}` mora biti unikatni identifikator projekta, v katerega želimo korak vstaviti. Če je vstavljanje v projekt uspešno, strežnik odjemalcu pošlje v odgovor kopijo dodanega objekta `Note`.

Storitev za projekte nato ta korak doda svoji zbirki korakov.

Pri vstavljanju slikovnih korakov je postopek vstavljanja podoben. Ko uporabnik odpre stran za dodajanje slikovnega koraka, se zažene privzeta aplikacija za kamero. Ko uporabnik fotografijo posname, se ta prikaže na strani z vnosnim obrazcem (slika 3.20). Slika se na mobilni napravi hrani v mapi ”Pictures/{naslov projekta}”, v uporabnikovem domačem direktoriju. Uporabnik ima na vnosnem obrazcu možnost sliko ponovno posneti ali jo odstraniti.

Uporabnik mora v vnosnem obrazcu podati naslov koraka (`Title`), opis koraka (`Text`) ter trajanje v urah, minutah in sekundah.

Uporabnik mato pritisniti gumb za vstavljanje slikovnega koraka v projekt. Storitev za upravljanje s projektom fotografijo zakodira v znakovni niz po metodi Base64. Storitev za komunikacijo s strežnikom zakodirano fotografijo in novo ustvarjen objekt razreda `Note` vključi v objekt tipa `UploadImageRequest`.

```
UploadImageRequest {  
    Note note;  
    string ImageString;  
}
```

Novo nastali zahtevek `UploadImageRequest` se nato avtorizira z JWT avtorizacijskim žetonom in pošlje na URL ”/projects/{id}/addimage”. Polje `{id}` mora biti unikatni identifikator projekta, v katerega želimo korak vstaviti. Če je vstavljanje v projekt uspešno, strežnik odjemalcu pošlje v odgovor kopijo pravkar vstavljenega objekta `Note`.

## Uporaba kamere v Xamarin

Za uporabo kamere v ogrodju Xamarin smo uporabili vtičnik `Xam.Plugin.Media` [11]. Vtičnik je licensiran pod MIT licenco.

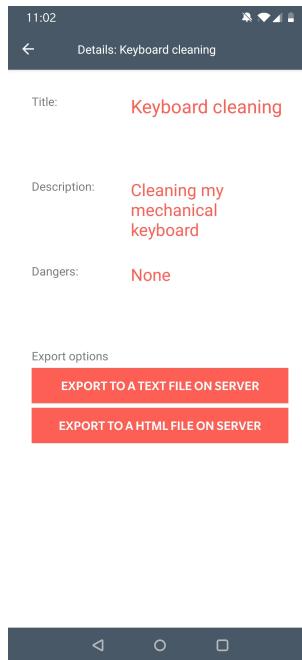
Zanj smo se odločili, saj podpira zajem fotografij s privzeto aplikacijo za zajemanje fotografij. Izbiramo lahko tudi kvaliteto zajete fotografije.

Pri zajemu slike najprej preverimo, ali imamo zadostne pravice. Imeti moramo pravice za dostop do shrambe, dostop do fotografij in dostop do kamere.

Fotografijo nato zajamememo z metodo `TakePictureAsync`. Kreirano fotografijo nato preberemo in znotraj aplikacije hranimo v objektu razreda `ImageSource`. Fotografijo v tej obliki lako prikažemo v XAML z grafičnim elementom `Image`. Lahko jo tudi zakodiramo v znakovni niz in pošljemo na strežnik.

### 3.6.7 Izvoz projekta

Uporabnik lahko projekt izvozi v človeško berljivo obliko na strežnik v dva različna formata: tekstovno datoteko (.txt) in HTML dokument.



Slika 3.21: Stran za izvoz projektov

Če želi uporabnik izvoziti projekt kot tekstovno datoteko, storitev za komunikacijo s strežnikom pošlje POST zahtevek na URL ”/project/{id}/export/text”. Če želi uporabnik izvoziti projekt kot HTML dokument, storitev za komunikacijo s strežnikom pošlje POST zahtevek na URL ”/project/{id}/export/html”.

## 3.7 Testiranje sistema OpenReport

### 3.7.1 Testne situacije

Pripravili smo štiri testne situacije.

1. Izdelava testnega opisa tehnološkega postopka **brez uporabe glasovnega asistenta**
  - (a) brez premora med koraki in
  - (b) z 90 sekundnim premorom med koraki.

## 2. Izdelava testnega opisa tehnološkega postopka z uporabo glasovnega asistenta

- (a) brez premora med koraki in
- (b) z 90 sekundnim premorom med koraki.

Z 90 sekundnim premorom smo simulirali delo na izdelku. Med premorom vpišemo testne podatke v obrazec za dodajanje koraka. Po 90 sekundnem premoru korak vstavimo v poročilo.

### Testno poročilo

Testno poročilo mora vsebovati 4 tekstovne korake in enega slikovnega. Pri situacijah z uporabo Alexe, mora uporabnik obrazce za korake odpreti z glasovnim ukazom.

### 3.7.2 Metoda testiranja in vpliv na hipotezo

Testne situacije brez premorov so služile za preverjanje hitrosti dela. Hipotezo bomo potrdili, če bomo pisanje testnega poročila končali prej z *uporabo Alexe* kot brez nje.

Če bo uporaba Alexe podaljšala čas pisanja, bomo hipotezo deloma ovrgli in prešli na drugo stopnjo testiranja. V drugi stopnji dodamo prejšnjemu postopku 90 sekundni premor med koraki. Ta premor simulira delo na izdelku.

Če bomo v tej situaciji poročilo z Alexo pisali enako hitro kot brez Alexe, hipoteza ostane delno ovržena. Če bo pisanje z Alexo počasnejše, kljub premorom, pa se bo hipotezo v celoti ovrglo.

### Testni koraki

Testni koraki simulirajo delo na delovnem mestu, kjer se moramo za uporabo mobilnega telefona oddaljiti od delovnega mesta.

Testni korak za dodajanje tekstovnega koraka je potekal tako:

1. premik do mobilnega telefona (3 metre),

2. odpiranje obrazca za dodajanje tekstovnega koraka,
3. vpis besedila "Test step text",
4. nastavitev trajanja na 10 minut,
5. vstavljanje koraka,
6. odmik od mobilnega telefona (3 metre).

Testni korak za dodajanje slikovnega koraka je potekal tako:

1. premik do mobilnega telefona (3 metre),
2. odpiranje obrazca za dodajanje slikovnega koraka,
3. zajem slike,
4. vpis besedila "Test step text",
5. nastavitev trajanja na 10 minut,
6. vstavljanje koraka,
7. odmik od mobilnega telefona (3 metre).

Pri situacijah z uporabo Amazon Alexe smo glasovne ukaze začeli izgovarjati, ko smo se začeli premikati proti mobilni napravi.

### **3.7.3 Rezultati testiranja in ugotovitve**

	<b>brez premorov</b>	<b>z 90 sekundnimi premori</b>
<b>brez Alexe</b>	1 minuta 51 sekund	7 minut 30 sekund
<b>z Alexo</b>	2 minuti 31 sekund	7 minut 30 sekund

Meritve časa pisanja poročil so pokazale, da je uporaba sistema OpenReport brez Amazon Alexe hitrejša v situacijah brez 90 sekundnih premorov. To našo hipotezo delno ovrže. V situacijah z 90 sekundnimi premori, je zapis korakov s pomočjo Amazon Alexe trajal enako dolgo kot brez nje. To našo hipotezo pusti delno ovrženo.

Testiranje zaključujemo z ugotovitvijo, da v situacijah brez premorov, Amazon Alexa dela ni pospešila. Hkrati pa smo ugotovili, da ga v testni situaciji s premori ni občutno ovirala.

Iz uporabniškega stališča prinese uporaba glasovnega asistenta v proces interakcije z računalniškim sistemom novo dimenzijo. V našem primeru smo lahko zahtevo za odpiranje vnosnega obrazca v mobilni aplikaciji izvedli že na poti do naprave.

### 3.8 Evalvacija funkcionalnosti

Primerjali smo pisanje opisa tehnološkega postopka na list papirja, s pisarniškim programom in s sistemom OpenReport. Sistem OpenReport smo testirali z uporabo Amazon Alexe in brez uporabe Amazon Alexe.

Mobilna aplikacija in strežnik sta izpolnila pričakovanja. Možnost pisanja tekstovnih korakov je smiselna za manjše opombe. Uporaba slikovnih korakov je smiselna za večje oporne točke pri delu.

Možnost izvoza projekta omogoča tudi prenos podatkov v druge računalniške programe.

Hitrost uporabe sistema se z uporabo glasovnega asistenta ni izboljšala.

Pod pričakovanji se je izkazalo narekovanje besedila glasovnemu asistentu. Razpoznavanje glasu pri „Skill-ih“ za asistenta Amazon Alexa namreč deluje najbolje, ko ima stavek veliko stopnjo ujemanja z definirano frazo.

- Alexa je ukaz ”Take a picture“ pravilno razpoznala v 18 od 20 testnih izgovorjav.

- Ukaz "Create a text note" je pravilno razpoznala v 17 od 20 testnih izgovorjav.
- Ukaz "note this is a voice note" je pravilno razpoznala v 3 od 20 testnih izgovorjav.
- Ukaza "note backplate removal takes about three minutes" ni pravilno razpoznala v nobeni od 20 testnih izgovorjav.

Dodajanje slikovnih in besedilnih korakov, s pomočjo glasovnega ukaza, se je izkazalo za delno uspešno. Uporabno je bilo predvsem, ko mobilnega telefona med delom nismo imeli na dosegu roke. Kljub temu je problem prikazovalo dejstvo, da je izgovorjava invokacijske fraze, fraz namer in čakanje na obdelavo zahtev, trajalo v povprečju 18 sekund.

Odklepanje mobilnega telefona, ki ga imamo v dosegu rok in ga lahko dosežemo brez presedanja ter odpiranje forme za dodajanje korakov je trajalo v povprečju 4 sekunde. Odklepanje mobilnega telefona, ki je oddaljen tri metre ter odpiranje forme za dodajanje korakov je trajalo povprečju 8 sekund.

# Poglavlje 4

## Zaključek

Sistem, razvit v sklopu diplomske naloge zajema funkcionalosti, ki uporabniku omogočajo pisanje in urejanje delavnih dnevnikov. Sestavlja ga strežnik, mobilna aplikacija in glasovni asistent Amazon Alexa. Uporabnik lahko v delavnike dodaja preproste tekstovne korake. Z uporabo mobilne aplikacije lahko zajame fotografije in jih v delavniški dnevnik vstavi kot slikovno gradivo.

Ta sistem je služil kot primer sistema, ki ga želimo izboljšati z uporabo glasovnega asistenta. Naš namen je bil pospešiti delo z glasovnim asistentom Amazon Alexo. Implementirali smo odpiranje kamere in obrazca za dodajanje korakov z glasovnim asistentom. Kljub uspešni implementaciji, je bil čas, potreben za izgovorjavo in obdelavo glasovnih ukazov, predolg, da bi poskus izboljšave bil popolnoma uspešen. Za neuspešno se je izkalo glasovno narekovanje vsebine koraka delavnih dnevnika. Z Amazon Alexo ni bilo mogoče razpoznati daljših, variirajočih glasovnih ukazov.

V možne izboljšave sistema smo na prvo mesto uvrstili preizkus drugih glasovnih asistentov, npr. Google Assistant [7]. Večjo učinkovitost bi lahko dosegli z asistentom, ki bi lahko razpoznał ukaze z eno samo frazo in ne kombinacijo dveh, kot Amazon Alexa. Druga smiselna izboljšava je izvoz delavnih dnevnikov v več različnih standardnih formatov, kot so npr. XSL ali CSV.

Mobilna aplikacija in strežniški program sta zadovoljila pričakovanja in lahko že v trenutnem stanju služita kot referenca za primerjavo učinkovitosti Amazon Alexe in drugih glasovnih asistentov.

Kljub ovrženi hipotezi pa je očitno, da je tehnologija glasovnih asistentov zelo primeren način za nadzor računalniških sistemov na daljavo. Poleg tega ponuja veliko možnosti za slepe, slabovidne in gibalno omejene posameznike, ki imajo težave z interakcijo z računalnikom.





# Literatura

- [1] *.NET ogrodje*. Dosegljivo: <https://dotnet.microsoft.com/>. [Dostopano: 18. 8. 2020].
- [2] *Amazon Alexa*. Dosegljivo: <https://www.amazon.com/b?ie=UTF8&node=17934671011>. [Dostopano: 12. 8. 2020].
- [3] Jonas Austerjost in sod. „Introducing a virtual assistant to the lab: A voice user interface for the intuitive control of laboratory instruments“. V: *SLAS TECHNOLOGY: Translating Life Sciences Innovation* 23.5 (2018), str. 476–482.
- [4] *Delavniki dnevnik kot izobraževalni pripomoček*. [Dostopano: 18. 8. 2020].
- [5] *Funkcionalnosti AWS Lambda*. Dosegljivo: <https://aws.amazon.com/lambda/features/>. [Dostopano: 12. 8. 2020].
- [6] *Funkcionalnosti AWS SQS*. Dosegljivo: <https://aws.amazon.com/sqs/features/>. [Dostopano: 12. 8. 2020].
- [7] *Google Assistant*. Dosegljivo: <https://assistant.google.com/>. [Dostopano: 18. 8. 2020].
- [8] *LibreOffice Writer*. Dosegljivo: <https://www.libreoffice.org/discover/writer/>. [Dostopano: 12. 8. 2020].
- [9] *Shema MVVM pristopa*. Dosegljivo: <https://docs.devexpress.com/WPF/Images/winforms-mvvm-common-mvvm-scheme118014.png>. [Dostopano: 18. 8. 2020].

- [10] *Uvod v Alexa "Skill-e"*. Dosegljivo: <https://dotnet.microsoft.com/apps/xamarin>. [Dostopano: 12. 8. 2020].
- [11] *Xam.Plugin.Media*. Dosegljivo: <https://github.com/jamesmontemagno/MediaPlugin>. [Dostopano: 12. 8. 2020].
- [12] *Xamarin*. Dosegljivo: <https://dotnet.microsoft.com/apps/xamarin>. [Dostopano: 12. 8. 2020].